IN-34
190207
~~[illegible]~~
99P

NASA Technical Memorandum 106340

Proteus Three-Dimensional Navier-Stokes Computer Code–Version 1.0

Volume 2–User's Guide

Charles E. Towne, John R. Schwab, and Trong T. Bui
Lewis Research Center
Cleveland, Ohio

October 1993

NASA

CONTENTS

PRINCIPAL NOTATION

SYMBOLS

Unless specified otherwise, all variables are nondimensional.

Symbol	Definition
c_p, c_v	Specific heats at constant pressure and volume.
C_p	Static pressure coefficient.
E_T	Total energy per unit volume.
g_c	Proportionality constant in Newton's second law.
h_T	Stagnation enthalpy per unit mass.
k	Effective thermal conductivity coefficient.
L_r	Dimensional reference length.
M	Mach number.
n	Time level.
N_1, N_2, N_3	Number of grid points in the ξ, η, and ζ directions.
p	Static pressure.
Pr_l	Laminar Prandtl number.
$\mathbf{Q}$	Vector of dependent variables in the Cartesian coordinate form of the governing equations.
R	Residual.
R	Gas constant.
Re_r	Reference Reynolds number.
T	Static temperature.
u, v, w	Velocities in the Cartesian x, y, and z directions.
x, y, z	Cartesian coordinates.
γ	Ratio of specific heats, c_p/c_v.
$\varepsilon_E^{(2)}$, $\varepsilon_E^{(4)}$	Second- and fourth-order explicit artificial viscosity coefficients in constant coefficient model.
ε_I	Implicit artificial viscosity coefficient.
θ_1, θ_2, θ_3	Parameters determining type of time differencing used.
κ_2, κ_4	Constants in nonlinear coefficient artificial viscosity model.
μ	Viscosity coefficient.
ξ, η, ζ	Computational coordinate directions.
ρ	Static density.
τ	Computational time.

SUBSCRIPTS

Subscript	Definition
i, j, k	Denotes grid location in ξ, η, and ζ directions.
n	Denotes dimensional normalizing condition.
r	Denotes dimensional reference condition.
T	Denotes total, or stagnation, value.

SUPERSCRIPTS

Superscript	Definition
n	Denotes time level.
—	Overbar; denotes dimensional value.

PROTEUS THREE-DIMENSIONAL NAVIER-STOKES COMPUTER CODE - VERSION 1.0

Volume 2 - User's Guide

Charles E. Towne, John R. Schwab, Trong T. Bui

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio

SUMMARY

A computer code called *Proteus* has been developed to solve the three-dimensional, Reynolds-averaged, unsteady compressible Navier-Stokes equations in strong conservation law form. The objective in this effort has been to develop a code for aerospace propulsion applications that is easy to use and easy to modify. Code readability, modularity, and documentation have been emphasized.

The governing equations are written in Cartesian coordinates and transformed into generalized nonorthogonal body-fitted coordinates. They are solved by marching in time using a fully-coupled alternating-direction-implicit solution procedure with generalized first- or second-order time differencing. The boundary conditions are also treated implicitly, and may be steady or unsteady. Spatially periodic boundary conditions are also available. All terms, including the diffusion terms, are linearized using second-order Taylor series expansions. Turbulence is modeled using either an algebraic or two-equation eddy viscosity model.

The program contains many operating options. The thin-layer or Euler equations may be solved as subsets of the Navier-Stokes equations. The energy equation may be eliminated by the assumption of constant total enthalpy. Explicit and implicit artificial viscosity may be used to damp pre- and post-shock oscillations in supersonic flow and to minimize odd-even decoupling caused by central spatial differencing of the convective terms in high Reynolds number flow. Several time step options are available for convergence acceleration, including a locally variable time step and global time step cycling. Simple Cartesian or cylindrical grids may be generated internally by the program. More complex geometries require an externally generated computational coordinate system.

The documentation is divided into three volumes. Volume 1 is the Analysis Description, and presents the equations and solution procedure used in *Proteus*. It describes in detail the governing equations, the turbulence model, the linearization of the equations and boundary conditions, the time and space differencing formulas, the ADI solution procedure, and the artificial viscosity models. Volume 2, the current volume, is the User's Guide, and contains information needed to run the program. It describes the program's general features, the input and output, the procedure for setting up initial conditions, the computer resource requirements, the diagnostic messages that may be generated, the job control language used to run the program, and several test cases. Volume 3 is the Programmer's Reference, and contains detailed information useful when modifying the program. It describes the program structure, the Fortran variables stored in common blocks, and the details of each subprogram.

A two-dimensional/axisymmetric version of *Proteus* code also exists, and was originally released in late 1989.

INTENTIONALLY BLANK

1.0 INTRODUCTION

Much of the effort in applied computational fluid dynamics consists of modifying an existing program for whatever geometries and flow regimes are of current interest to the researcher. Unfortunately, nearly all of the available non-proprietary programs were started as research projects with the emphasis on demonstrating the numerical algorithm rather than ease of use or ease of modification. The developers usually intend to clean up and formally document the program, but the immediate need to extend it to new geometries and flow regimes takes precedence.

The result is often a haphazard collection of poorly written code without any consistent structure. An extensively modified program may not even perform as expected under certain combinations of operating options. Each new user must invest considerable time and effort in attempting to understand the underlying structure of the program if intending to do anything more than run standard test cases with it. The user's subsequent modifications further obscure the program structure and therefore make it even more difficult for others to understand.

The *Proteus* three-dimensional Navier-Stokes computer program is a user-oriented and easily-modifiable flow analysis program for aerospace propulsion applications. Readability, modularity, and documentation were primary objectives during its development. The entire program was specified, designed, and implemented in a controlled, systematic manner. Strict programming standards were enforced by immediate peer review of code modules; Kernighan and Plauger (1978) provided many useful ideas about consistent programming style. Every subroutine contains an extensive comment section describing the purpose, input variables, output variables, and calling sequence of the subroutine. With just three clearly-defined exceptions, the entire program is written in ANSI standard Fortran 77 to enhance portability. A master version of the program is maintained and periodically updated with corrections, as well as extensions of general interest (e.g., turbulence models.)

The *Proteus* program solves the unsteady, compressible, Reynolds-averaged Navier-Stokes equations in strong conservation law form. The governing equations are written in Cartesian coordinates and transformed into generalized nonorthogonal body-fitted coordinates. They are solved by marching in time using a fully-coupled alternating-direction-implicit (ADI) scheme with generalized time and space differencing (Briley and McDonald, 1977; Beam and Warming, 1978). Turbulence is modeled using either the Baldwin and Lomax (1978) algebraic eddy-viscosity model or the Chien (1982) two-equation model. All terms, including the diffusion terms, are linearized using second-order Taylor series expansions. The boundary conditions are treated implicitly, and may be steady or unsteady. Spatially periodic boundary conditions are also available.

The program contains many operating options. The thin-layer or Euler equations may be solved as subsets of the Navier-Stokes equations. The energy equation may be eliminated by the assumption of constant total enthalpy. Explicit and implicit artificial viscosity may be used to damp pre- and post-shock oscillations in supersonic flow and to minimize odd-even decoupling caused by central spatial differencing of the convective terms in high Reynolds number flow. Several time step options are available for convergence acceleration, including a locally variable time step and global time step cycling. Simple grids may be generated internally by the program; more complex geometries require external grid generation, such as that developed by Chen and Schwab (1988).

The documentation is divided into three volumes. Volume 1 is the Analysis Description, and presents the equations and solution procedure used in *Proteus*. It describes in detail the governing equations, the turbulence model, the linearization of the equations and boundary conditions, the time and space differencing formulas, the ADI solution procedure, and the artificial viscosity models. Volume 2, the current volume, is the User's Guide, and contains information needed to run the program. It describes the program's general features, the input and output, the procedure for setting up initial conditions, the computer resource requirements, the diagnostic messages that may be generated, the job control language used to run the program, and several test cases. Volume 3 is the Programmer's Reference, and contains detailed information useful when modifying the program. It describes the program structure, the Fortran variables stored in common blocks, and the details of each subprogram.

A two-dimensional/axisymmetric version of *Proteus* code also exists, and was originally released in late 1989 (Towne, Schwab, Benson, and Suresh, 1990).

The authors would like to acknowledge the significant contributions made by their co-workers. Tom Benson provided part of the original impetus for the development of *Proteus*, and did the original coding of the block tri-diagonal inversion routines. Simon Chen did the original coding of the Baldwin-Lomax turbulence model, and consulted in the implementation of the nonlinear coefficient artificial viscosity model. William Kunik developed the original code for computing the metrics of the generalized nonorthogonal grid transformation. Frank Molls has created a separate diagonalized version of the code. Ambady Suresh did the original coding for the second-order time differencing and for the nonlinear coefficient artificial viscosity model. These people, along with Dick Cavicchi, Julie Conley, Jason Solbeck, and Pat Zeman, have also run many debugging and verification cases.

2.0 GENERAL DESCRIPTION

In this section the basic characteristics and capabilities of the *Proteus* code are described in general. More detailed descriptions can be found in other sections of this manual or in Volumes 1 and 3.

2.1 ANALYSIS

Proteus 3-D solves the three-dimensional unsteady compressible Navier-Stokes equations. The equations are solved in fully conservative form. As subsets of these equations, options are available to solve the Euler equations or the thin-layer Navier-Stokes equations. An option is also available to eliminate the energy equation by assuming constant total enthalpy. The governing equations are described in detail in Section 2.0 of Volume 1.

The equations are solved by marching in time using the generalized time differencing of Beam and Warming (1978). The method may be either first- or second-order accurate in time, depending on the choice of time differencing parameters. Second-order central differencing is used for all spatial derivatives. The time and space differencing formulas are presented in Sections 3.0 and 5.0 of Volume 1. Nonlinear terms are linearized using second-order Taylor series expansions in time, as described in Section 4.0 of Volume 1. The resulting difference equations are solved using an alternating-direction implicit (ADI) technique, with Douglas-Gunn type splitting as written by Briley and McDonald (1977). The boundary conditions are also treated implicitly.

Artificial viscosity, or smoothing, is normally added to the solution algorithm to damp pre- and post-shock oscillations in supersonic flow, and to prevent odd-even decoupling due to the use of central differences in convection-dominated regions of the flow. Implicit smoothing and two types of explicit smoothing are available in *Proteus*. The implicit smoothing is second order with constant coefficients. For the explicit smoothing the user may choose a constant coefficient second- and/or fourth-order model (Steger, 1978), or a nonlinear coefficient mixed second- and fourth-order model (Jameson, Schmidt, and Turkel, 1981). The nonlinear coefficient model was designed specifically for flow with shock waves. The artificial viscosity models are described in detail in Section 8.0 of Volume 1.

The equations are fully coupled, leading to a system of equations with a block tridiagonal coefficient matrix that can be solved using the block matrix version of the Thomas algorithm. Because this algorithm is recursive, the source code cannot be vectorized in the ADI sweep direction. However, it is vectorized in one of the non-sweep directions, leading to an efficient implementation of the algorithm. The solution algorithm is described in detail in Section 7.0 of Volume 1.

2.2 GEOMETRY AND GRID SYSTEM

The equations solved in *Proteus* were originally written in a Cartesian coordinate system, then transformed into a general nonorthogonal computational coordinate system as described in Section 2.3 of Volume 1. The code is therefore not limited to any particular type of geometry or coordinate system. The only requirement is that body-fitted coordinates must be used. In general, the computational coordinate system for a particular geometry must be created by a separate coordinate generation code and stored in an unformatted file that *Proteus* can read. However, simple Cartesian and cylindrical coordinate systems are built in.

The equations are solved at grid points that form a computational mesh within this computational coordinate system. Note that a distinction is being made between the terms *computational coordinate system* and *computational mesh*. The *computational coordinate system* refers to the (ξ,η,ζ) system in which the governing equations are written. It is determined by supplying a series of points whose Cartesian (x,y,z) coordinates are specified, either by reading them from a file or through one of the analytically defined coordinate systems built into subroutine GEOM. The *computational mesh* consists of grid points distributed along lines in the computational coordinate directions. These points may differ in number and location from those used to determine the computational coordinate system. The number of grid points in each direction in the computational mesh is specified by the user. The location of these grid points can be varied by packing them at either or both boundaries in any coordinate direction. The transformation metrics and Jacobian are computed using finite differences in a manner consistent with the differencing of the governing equations.

2.3 FLOW AND REFERENCE CONDITIONS

As stated earlier, the equations solved by *Proteus* are for compressible flow. Incompressible conditions can be simulated by running at a Mach number of around 0.1. Lower Mach numbers may lead to numerical problems. The flow can be laminar or turbulent. The gas constant $\bar{R}$ is specified by the user, with the value for air as the default. The specific heats c_p and c_v, the molecular viscosity μ, and the thermal conductivity k can be treated as constants or as functions of temperature. The empirical formulas used to relate these properties to temperature are contained in subroutine FTEMP, and can easily be modified if necessary. The perfect gas equation of state is used to relate pressure, density, and temperature. This equation is contained in subroutine EQSTAT, which could also be easily modified if necessary. All equations and variables in the program are nondimensionalized by normalizing values derived from reference conditions specified by the user, with values for sea level air as the default.

2.4 BOUNDARY CONDITIONS

The easiest way to specify boundary conditions in *Proteus* is by specifying the type of boundary (i.e., no-slip adiabatic wall, subsonic inflow, periodic, etc.). The program will then select an appropriate set of conditions for that boundary. For most applications this method should be sufficient. If necessary, however, the user may instead set the individual boundary conditions on any or all of the six computational boundaries.

A variety of individual boundary conditions are built into the *Proteus* code, including: (1) specified values and/or gradients of Cartesian velocities u, v, and w, normal velocity V_n, coordinate direction velocities V_ξ, V_η, and V_ζ, pressure p, temperature T, and density ρ; (2) specified values of total pressure p_T, total temperature T_T, and flow angles; and (3) linear extrapolation. Another useful boundary condition is a "no change from initial condition" option for u, v, w, p, T, ρ, p_T, and/or T_T. Provision is also made for user-written boundary conditions using subroutines BCF and BCFLIN. Specified gradient boundary conditions may be in the direction of the coordinate line intersecting the boundary or normal to the boundary, and may be computed using two-point or three-point difference formulas. For all of these conditions, the same type and value may be applied over the entire boundary surface, or a point-by-point distribution may be specified. Unsteady and time-periodic boundary conditions are allowed when applied over the entire boundary. The boundary conditions available in *Proteus* are described in detail in Section 3.1.7.

2.5 INITIAL CONDITIONS

Initial conditions are required throughout the flow field to start the time marching procedure. For unsteady flows they should represent a real flow field. A converged steady-state solution from a previous run would be a good choice. For steady flows, the ideal initial conditions would represent a real flow field that is close to the expected final solution.

The initial conditions are set up in subroutine INIT. A default version of INIT is built into *Proteus* that specifies uniform flow with constant flow properties everywhere in the flow field. These conditions, of course, do represent a solution to the governing equations, and for many problems may help minimize starting transients in the time marching procedure. However, realistic initial conditions that are closer to the expected final solution should lead to quicker convergence.

The best choice for initial conditions, therefore, will vary from problem to problem. For this reason *Proteus* does not include a more generalized routine for setting up initial conditions. Instead, the user should supply his or her own version of subroutine INIT to set up the initial conditions for the particular flow being computed. Details on the Fortran variables to be specified by INIT may be found in Section 5.1.

2.6 TIME STEP SELECTION

Several different options are available for choosing the time step $\Delta\tau$, and for modifying it as the solution proceeds. $\Delta\tau$ may be specified directly, or through a value of the Courant-Friedrichs-Lewy (CFL) number. When specifying a CFL number, the time step $\Delta\tau$ may be either *global* (i.e., constant in space) based on the minimum CFL limit, or *local* (i.e., varying in space) based on the local CFL limit. For unsteady time-accurate flows global values should be used, but for steady flows using local values may lead to faster convergence. Options are available to increase or decrease $\Delta\tau$ as the solution proceeds based on the change in the dependent variables. An option is also available to cycle $\Delta\tau$ between two values in a logarithmic progression over a specified number of time steps. The various time step options are described in detail in Section 3.1.9.

2.7 CONVERGENCE

Five options are currently available for determining convergence. The user specifies a convergence criterion ε for each of the governing equations. Then, depending on the option chosen, convergence is based on: (1) the absolute value of the maximum change in the conservation variables ΔQ_{max} over a single time step; (2) the absolute value of the maximum change ΔQ_{max} averaged over a specified number of time steps; (3) the L_2 norm of the residual for each equation; (4) the average residual for each equation; or (5) the maximum residual for each equation. These criteria are defined in Section 4.1.6.

2.8 INPUT/OUTPUT

Input to *Proteus* is through a series of namelists[1] and, in general, an unformatted file containing the computational coordinate system. All of the input parameters have default values and only need to be specified by the user if a different value is desired. Reference conditions may be specified in either English or SI units. The namelist and coordinate system input are described in Section 3.0. A restart option is also available, in which the computational mesh and the initial flow field are read from unformatted restart files created during an earlier run. The use of the restart option is described in Sections 3.1.3 and 5.3.

The standard printed output available in *Proteus* includes an echo of the input, boundary conditions, normalizing and reference conditions, the computed flow field, and convergence information. The user controls exactly which flow field parameters are printed, and at which time levels and grid points. Several debug options are also available for detailed printout in various parts of the program. The printed output is described in Section 4.1.

In addition to the printed output, several unformatted files can be written for various purposes. The first is an auxiliary file used for post-processing, usually called a plot file, that can be written at convergence or after the last time step if the solution does not converge. Plot files can be written for the NASA Lewis plotting program CONTOUR or the NASA Ames plotting program PLOT3D. If PLOT3D is to be used, two unformatted files are created, an XYZ file containing the computational mesh and a Q file containing the computed flow field. The plot files are described in detail in Section 4.2. Another unformatted file written by *Proteus* contains detailed convergence information. This file is automatically incremented each time the solution is checked for convergence, and is used to generate the convergence history printout and with Lewis-developed post-processing plotting routines. The contents of the convergence history file are presented in Section 4.3. And finally, two unformatted files may be written at the end of a calculation that may be used to restart the calculation in a later run. One of these contains the computational mesh, and the other the computed flow field. The contents of the restart files are described in detail in Section 4.4.

2.9 TURBULENCE MODELS

For turbulent flow, *Proteus* solves the Reynolds time-averaged Navier-Stokes equations, with turbulence modeled using either the Baldwin and Lomax (1978) algebraic eddy-viscosity model or the Chien (1982) two-equation model. Both models are described in detail in Section 9.0 of Volume 1.

[1] It should be noted that namelist input is not part of ANSI standard Fortran 77, but is nevertheless available with most Fortran compilers. See Section 2.3.1 of Volume 3 for a discussion of possible computer-dependent features in the *Proteus* code.

2.9.1 Baldwin-Lomax Model

The Baldwin-Lomax model may be applied to either wall-bounded flows or to free turbulent flows. For wall-bounded flows, the model is a two-layer model. In the inner region, in addition to the Baldwin-Lomax formulation, an alternate expression first presented by Spalding (1961), and later by Kleinstein (1967), may be used. For flows in which more than one boundary is a solid surface, the nearest boundary is used. The turbulent thermal conductivity coefficient k_t is computed using Reynolds analogy.

2.9.2 Chien k-ε Model

With the Chien two-equation model, partial differential equations are solved for the turbulent kinetic energy k and the turbulent dissipation rate ε. These equations are lagged in time and solved separately from the Navier-Stokes equations. An LU factorization algorithm is used to solve the k-ε equations.

Since the Chien two-equation model is a low Reynolds number formulation, the k-ε equations are solved in the near-wall region. No additional approximations are needed. Boundary conditions that may be used include: (1) no change from initial or restart conditions for k and ε; (2) specified values and/or gradients of k and ε; and (3) linear extrapolation. Specified gradient boundary conditions are in the direction of the coordinate line intersecting the boundary, and may be computed using two-point or three-point difference formulas. For all of these conditions, the same type and value may be applied over the entire boundary surface, or a point-by-point distribution may be specified. Spatially periodic boundary conditions for k and ε may also be used. Unsteady boundary conditions are not available for the k-ε equations. However, unsteady flows can still be computed with the k-ε model using the unsteady boundary condition option for the mean flow quantities and appropriate boundary conditions for k and ε, such as specified gradients or linear extrapolation.

Initial conditions for k and ε are required throughout the flow field to start the time marching procedure. The best choice for initial conditions for the k-ε equations will vary from problem to problem, and the user may supply a subroutine, called KEINIT, that sets up the initial values for k and ε for the time marching procedure. Details on the Fortran variables to be specified by KEINIT may be found in Section 5.1. A version of KEINIT is built into *Proteus* that computes the initial k and ε values from a mean initial or restart flow field based on the assumption of local equilibrium (i.e., production equals dissipation.) Variations of that scheme have been found to be useful in computing the k-ε initial conditions for a variety of turbulent flows.

The time step used in the solution of the k-ε equations is normally the same as the time step used for the mean flow equations. However, the user can alter the time step for the k-ε equations, making it larger or smaller than the time step for the mean flow equations, by specifying a multiplication factor. The user can also specify the number of k-ε iterations per mean flow iteration.

3.0 INPUT DESCRIPTION

The standard input to the three-dimensional version of *Proteus* consists of a title line and several namelists. Additional input may be provided in the form of a pre-stored unformatted file containing the computational coordinate system. The calculation can also be started by reading the computational mesh and the initial flow field from restart files written during a previous run. This section describes only the standard input and the coordinate system file. The restart file contents and format are described in Section 4.4.

3.1 STANDARD INPUT

All of the standard input parameters have default values and do not need to be specified by the user unless some other value is desired. The type (real or integer) of the input parameters follows standard Fortran convention, unless stated otherwise (i.e., those starting with I, J, K, L, M, or N are integer, and the remainder are real.)[2] Note that in most, if not all, implementations of Fortran, namelist names and input start in character position 2 or higher in the input line. All of the input, except for namelist IC, is read in subroutine INPUT. Namelist IC is read in subroutine INIT.

3.1.1 Reference and Normalizing Conditions

Unless specified otherwise, all of the input parameters are specified in nondimensional form, with the appropriate *reference* condition as the nondimensionalizing factor. A few words explaining what we mean by *reference* conditions and *normalizing* conditions, and the differences between them, may be helpful at this point.

The *normalizing* conditions are, by definition, the conditions used in nondimensionalizing the governing equations, and are denoted by an *n* subscript. (See Section 2.0 of Volume 1.) These normalizing conditions are defined by six basic *reference* conditions, for length, velocity, temperature, density, viscosity, and thermal conductivity, which are specified by the user. Reference conditions are denoted by an *r* subscript. The normalizing conditions used in *Proteus* are listed in Table 3-1.

Note that for some variables, like pressure, the normalizing condition is dictated by the form of the governing equations once the six basic reference conditions are chosen. Unfortunately, some of these may not be physically meaningful or convenient for use in setting up input conditions. Therefore, some additional reference conditions are defined from the six user-supplied ones. The reference conditions are listed in Table 3-2.

To summarize, the *normalizing* conditions are used to nondimensionalize the governing equations. The average user need not be too concerned about these. The *reference* conditions are the ones used for nondimensionalization of all user-specified input and output parameters.[3]

3.1.2 Title

TITLE	A descriptive title, used on the printed output and in the CONTOUR plot file, up to 72 characters long. This is a character variable.

[2] Throughout this User's Guide, elements of the Fortran language, such as input variables and subprogram names, are printed in the text using uppercase letters. However, in most implementations, Fortran is case-insensitive. The *Proteus* source code itself is written in lowercase.

[3] Internal to the *Proteus* computer code itself, variables are generally nondimensionalized by the normalizing conditions. The reference conditions are used for input and output because they are usually more physically meaningful for the user.

3.1.3 Namelist RSTRT

The parameters in this namelist control the use of the restart option. The contents of the restart files are described in Section 4.4.

IREST	0 if no restart files are to be read or written. 1 to write restart files at the end of the calculation. 2 to read restart files for initial values, and to write restart files at the end of the calculation. For turbulent flow, the current calculation must use the same turbulence model as the previous calculation. Note that only the initial values and the computational mesh are read from restart files. The usual namelist input must still be read in. Of course, some input parameters, such as the reference conditions or those specifying the grid, must not be changed during a restart. 3 same as the IREST = 2 option, but for cases in which the previous calculation was either laminar or used an algebraic turbulence model, and the current calculation uses a two-equation turbulence model. Two restart files are written and/or read. One contains the computational mesh, and the other contains the mean flow field and the k and ε values (if a two-equation turbulence model is being used.) For IREST = 0 or 1, the initial mean flow field will be generated in subroutine INIT, and the initial k and ε values (if necessary) will be generated in subroutine KEINIT. The default value is 0.
NRQIN	Unit number for reading the restart flow field. The default value is 11.
NRQOUT	Unit number for writing the restart flow field. The default value is 12.
NRXIN	Unit number for reading the restart computational mesh. The default value is 13.
NRXOUT	Unit number for writing the restart computational mesh. The default value is 14.

3.1.4 Namelist IO

Printout Controls

The following parameters specify which variables are to be printed, and at what locations in both time and space.

IVOUT	An array of up to 50 elements specifying which variables are to be printed. The variables currently available for printing are listed and defined in Table 3-3.[4] The default values are 1, 2, 3, 20, 30, 40, 44*0, corresponding to printout of x, y, and z-velocity, and static density, pressure, and temperature.
IWOUT1	A 2-element array, given as IWOUT1(IBOUND), indicating whether or not various parameters are to be printed along the ξ boundaries. The subscript IBOUND = 1 or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. Valid values of IWOUT1(IBOUND) are: 0 for no printout. 1 for printout along the boundary, with normal derivatives computed using three-point one-sided differencing. −1 for printout along the boundary, with normal derivatives computed using two-point one-sided differencing.

[4] The definitions of k_l and k_t in Table 3-3 (IVOUT = 92 and 102) assume a constant turbulent Prandtl number is being specified in namelist TURB.

The default values are both 0.

IWOUT2 A 2-element array, given as IWOUT2(IBOUND), indicating whether or not various parameters are to be printed along the η boundaries. The subscript IBOUND = 1 or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. Valid values of IWOUT2(IBOUND) are:

0	for no printout.
1	for printout along the boundary, with normal derivatives computed using three-point one-sided differencing.
-1	for printout along the boundary, with normal derivatives computed using two-point one-sided differencing.

The default values are both 0.

IWOUT3 A 2-element array, given as IWOUT3(IBOUND), indicating whether or not various parameters are to be printed along the ζ boundaries. The subscript IBOUND = 1 or 2, corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. Valid values of IWOUT3(IBOUND) are:

0	for no printout.
1	for printout along the boundary, with normal derivatives computed using three-point one-sided differencing.
-1	for printout along the boundary, with normal derivatives computed using two-point one-sided differencing.

The default values are both 0.

The parameters printed via the IWOUT1, IWOUT2, and/or IWOUT3 input arrays are the Cartesian coordinates x, y, and z, the static pressure p, the skin friction coefficient c_f, the shear stress τ_w, the static temperature T, the heat transfer coefficient h, the heat flux q_w, and the Stanton number St. See Section 4.1 for the definitions of these parameters. Note that some of these parameters are meaningful only if the boundary is a solid wall. In general, using three-point one-sided differences to compute normal derivatives will be more accurate. However, for nearly-separated flows, it has been found that three-point differencing can give misleading negative values of τ_w and c_f. The parameters are printed at the boundary points specified by the input parameters IPRT1, IPRT2, and IPRT3, or IPRT1A, IPRT2A, and IPRT3A.

IDEBUG An array of up to 20 elements used to turn on additional printout, normally used for debugging purposes. Except where noted, set IDEBUG(I) = 1 for printout number I. For options 1 through 7, the input parameters IPRT1, IPRT2, and IPRT3, or IPRT1A, IPRT2A, and IPRT3A, determine the grid points at which the printout appears. Note that some of these options can generate *a lot* of output. Judicious use of the "IPRT" controls is recommended. The debug options currently available are as follows:

Number	Printout
1	Coefficient block submatrices and source term subvectors at time level n = IDEBUG(1) if IDEBUG(1) > 0, or at time levels $n \geq$ |IDEBUG(1)| if IDEBUG(1) < 0. This printout is done after the elimination of any off-diagonal boundary condition submatrices (subroutine BCELIM) and after any artificial viscosity has been added (subroutine AVISC1 or 2), but before any rearrangement of the elements in the boundary condition submatrices (subroutine FILTER.)

Number	Printout
2	Coefficient block submatrices and source term subvectors at time level $n =$ IDEBUG(2) if IDEBUG(2) > 0, or at time levels $n \geq \lvert$IDEBUG(2)$\rvert$ if IDEBUG(2) < 0. This printout is done after the elimination of any off-diagonal boundary condition submatrices (subroutine BCELIM), but before any artificial viscosity has been added (subroutine AVISC1 or 2) and before any rearrangement of the elements in the boundary condition submatrices (subroutine FILTER.)
3	Boundary condition coefficient block submatrices and source term subvectors at time level $n =$ IDEBUG(3) if IDEBUG(3) > 0, or at time levels $n \geq \lvert$IDEBUG(3)$\rvert$ if IDEBUG(3) < 0. This printout is done before the elimination of any off-diagonal boundary condition submatrices (subroutine BCELIM) and before any rearrangement of the elements in the boundary condition submatrices (subroutine FILTER.)
4	Boundary condition coefficient block submatrices and source term subvectors at time level $n =$ IDEBUG(4) if IDEBUG(4) > 0, or at time levels $n \geq \lvert$IDEBUG(4)$\rvert$ if IDEBUG(4) < 0. This printout is done after the elimination of any off-diagonal boundary condition submatrices (subroutine BCELIM) and after any rearrangement of the elements in the boundary condition submatrices (subroutine FILTER.)
5	Intermediate solutions $\mathbf{Q}^*$ and $\mathbf{Q}^{**}$ at time level $n =$ IDEBUG(5) if IDEBUG(5) > 0, or at time levels $n \geq \lvert$IDEBUG(5)$\rvert$ if IDEBUG(5) < 0.
6	Final solution $\mathbf{Q}^n$ after the last ADI sweep at time level $n =$ IDEBUG(6) if IDEBUG(6) > 0, or at time levels $n \geq \lvert$IDEBUG(6)$\rvert$ if IDEBUG(6) < 0.
7	Cartesian coordinates, metric coefficients, and inverse of the grid transformation Jacobian computed in subroutine METS.

The default values are all 0.

IUNITS	0 for input and output in English units. 1 for input and output in SI units. The default value is 0.
IPRT	Results are printed every IPRT'th time level. However, the initial and final flow fields are always printed. The default value is 1.
IPRTA	An array of up to 101 elements specifying the time levels at which results are to be printed. The initial conditions are at time level 1. If the calculation converges, or if the pressure or temperature is non-positive, the results are printed regardless of the value of IPRTA. If this parameter is specified, it overrides the value of IPRT. The default values are all 0.
IPRT1	Results are printed at every IPRT1'th grid point in the ξ direction. However, the results at the boundaries are always printed. The default value is 1.
IPRT2	Results are printed at every IPRT2'th grid point in the η direction. However, the results at the boundaries are always printed. The default value is 1.
IPRT3	Results are printed at every IPRT3'th grid point in the ζ direction. However, the results at the boundaries are always printed. The default value is 1.

IPRT1A	An array of up to N1 elements (see Namelist NUM) specifying the ξ indices at which results are to be printed. The indices must be specified in ascending order. If this parameter is specified, it overrides the value of IPRT1. The default values are all 0.
IPRT2A	An array of up to N2 elements (see Namelist NUM) specifying the η indices at which results are to be printed. The indices must be specified in ascending order. If this parameter is specified, it overrides the value of IPRT2. The default values are all 0.
IPRT3A	An array of up to N3 elements (see Namelist NUM) specifying the ζ indices at which results are to be printed. The indices must be specified in ascending order. If this parameter is specified, it overrides the value of IPRT3. The default values are all 0.
NHMAX	Maximum number of time levels allowed in the printout of the convergence history file (not counting the first two, which are always printed.) The default value is 100.

Plot File Controls

In addition to the printed output, files called plot files may be written for use by various post-processing routines. The following parameters specify the type of plot files to be written, and at what locations in both time and space. These plot files are described in greater detail in Section 4.2.

IPLOT

0 for no plot file.
1 to write results into an auxiliary file, in CONTOUR format, for later post-processing. If multiple time levels are to be written into the file, they will be stacked sequentially. The value of the time τ will not be written into the file.
2 to write results into auxiliary files, in PLOT3D/WHOLE format. Multiple time levels will be stacked sequentially,[5] with $\tau_{1,1,1}$ stored in the Q file header.[6]
3 to write results into auxiliary files, in PLOT3D/PLANES format. Multiple time levels will be stacked sequentially,[5] with $\tau_{1,1,1}$ stored in the Q file header.[6]

The default value is 0.

IPLT	Results are written into the plot file every IPLT'th time level. However, if IPLT > 0, the initial and final flow fields are automatically included in the file. If IPLT = 0, only the final flow field is written into the file. The default value is 0.
IPLTA	An array of up to 101 elements specifying the time levels at which results are to be written into the plot file. The initial conditions are at time level 1. If the calculation converges, or if the pressure or temperature is non-positive, the results are written into the plot file regardless of the value of IPLTA. If this parameter is specified, it overrides the value of IPLT. The default values are all 0.

Unit Numbers

The following parameters specify the Fortran unit numbers used for various input and output files. NIN, the unit number for reading the standard input file, is hardwired in the program as 5.

NOUT	Unit number for printing standard output. The default value is 6.

[5] The current version of PLOT3D does not work for multiple time levels, although future versions might.

[6] Note that with IDTAU = 5 or 6, τ will vary in space, and therefore $\tau_{i,j,k} \neq \tau_{1,1,1}$.

NGRID	Unit number for reading computational coordinate system file. The default value is 7.
NPLOTX	Unit number for writing XYZ file when using PLOT3D plot file format. The default value is 8.
NPLOT	Unit number for writing CONTOUR plot file, or for writing Q file when using PLOT3D format. The default value is 9.
NHIST	Unit number for writing convergence history file. The default value is 10.

3.1.5 Namelist GMTRY

Coordinate System Type

These parameters specify the type of flow domain being analyzed. Simple Cartesian or cylindrical configurations can be done automatically. For more complex geometries, the configuration is determined by reading a pre-stored coordinate file. Note however, that the number of grid points and their distribution can be changed by the parameters in namelist NUM.

NGEOM Flag used to specify type of computational coordinates. Currently coded are:

1 Cartesian (x-y-z) computational coordinates.
2 Cylindrical (r-θ-x) computational coordinates.
10 Get computational coordinates from coordinate system file. The contents of this file are described in Section 3.2.

The default value is 1.

Cartesian Computational Coordinates

The following parameters specify the size of the flow domain for the Cartesian coordinate option (NGEOM = 1). The computational (ξ,η,ζ) domain for this option is shown in physical (x,y,z) space in Figure 3.1. The points in the figure labeled "min" and "max" are the points ($x_{min}, y_{min}, z_{min}$) and ($x_{max}, y_{max}, z_{max}$), respectively.

XMIN	Minimum x-coordinate for Cartesian coordinate option. The default value is 0.0.
XMAX	Maximum x-coordinate for Cartesian coordinate option. The default value is 1.0.
YMIN	Minimum y-coordinate for Cartesian coordinate option. The default value is 0.0.
YMAX	Maximum y-coordinate for Cartesian coordinate option. The default value is 1.0.
ZMIN	Minimum z-coordinate for Cartesian coordinate option. The default value is 0.0.
ZMAX	Maximum z-coordinate for Cartesian coordinate option. The default value is 1.0.

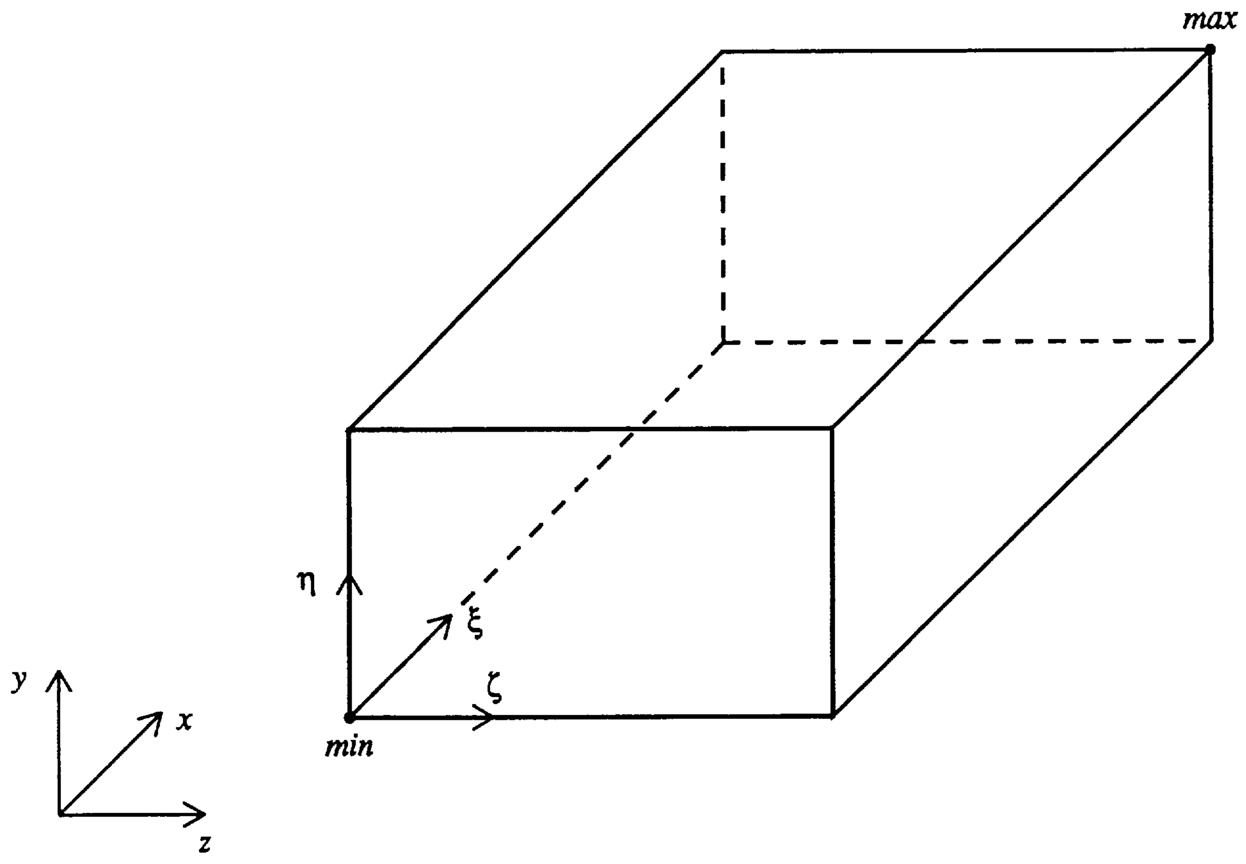

Figure 3.1 - Cartesian computational coordinates.

Cylindrical Computational Coordinates

The following parameters specify the size of the flow domain for the cylindrical coordinate option (NGEOM = 2). The computational (ξ,η,ζ) domain for this option is shown in physical (x,y,z) space in Figure 3.2. The points in the figure labeled "min" and "max" are the points $(r_{min}, \theta_{min}, x_{min})$ and $(r_{max}, \theta_{max}, x_{max})$, respectively.

RMIN	Minimum r-coordinate for cylindrical coordinate option. The default value is 0.0.
RMAX	Maximum r-coordinate for cylindrical coordinate option. The default value is 1.0.
THMIN	Minimum θ-coordinate in degrees for cylindrical coordinate option. The default value is 0.0.
THMAX	Maximum θ-coordinate in degrees for cylindrical coordinate option. The default value is 90.0.
XMIN	Minimum x-coordinate for cylindrical coordinate option. The default value is 0.0.
XMAX	Maximum x-coordinate for cylindrical coordinate option. The default value is 1.0.

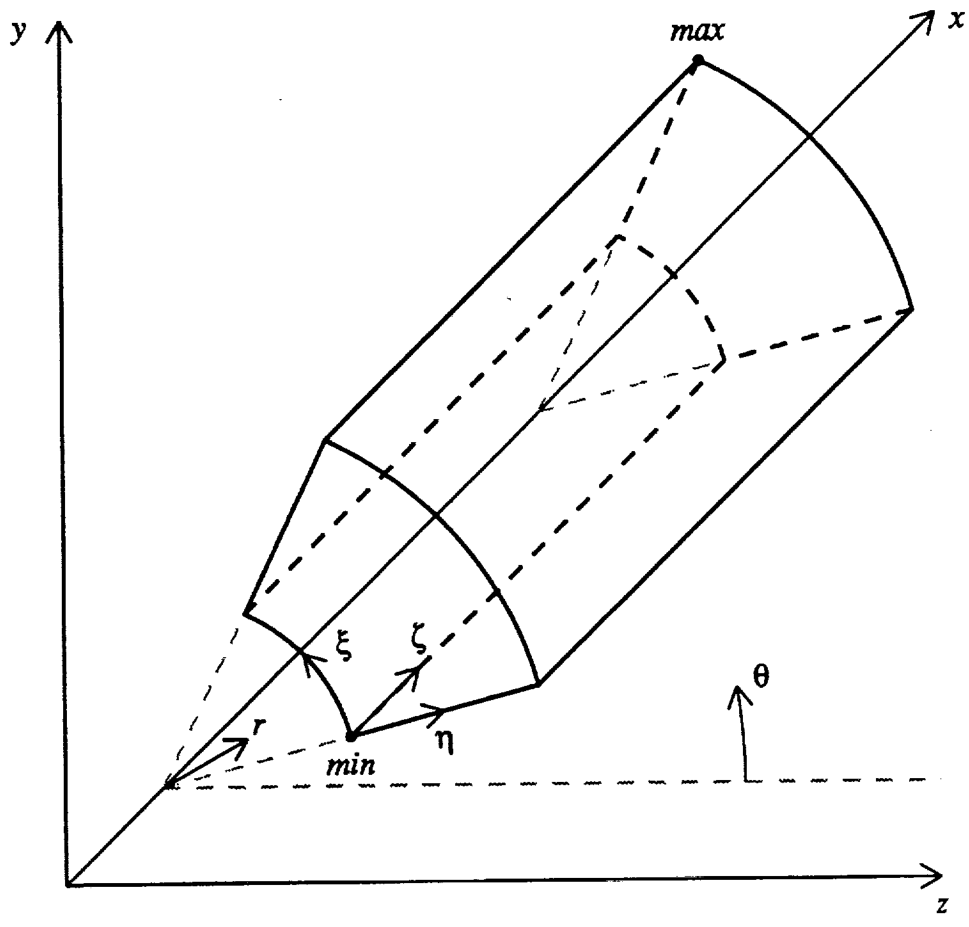

Figure 3.2 - Cylindrical computational coordinates.

3.1.6 Namelist FLOW

Control Flags

The following parameters are flags that specify the type of equations to be solved, and which variables are being supplied as initial conditions.

IEULER	0 for a full time-averaged Navier-Stokes calculation. 1 for an Euler calculation (i.e., neglecting all viscous and heat conduction terms.) The default value is 0.
ITHIN	A 3-element array, specified as ITHIN(IDIR), indicating whether or not the thin-layer option is to be used in direction IDIR. The subscript IDIR = 1, 2, or 3, corresponding to the ξ, η, and ζ directions, respectively. Valid values of ITHIN(IDIR) are: 0 to include second derivative viscous terms in direction IDIR. 1 to use the thin-layer option in direction IDIR. This does not decrease the execution time much, but may be useful if the grid in direction IDIR is not sufficiently dense to resolve second derivatives in that direction.

Only one of the three values may be set equal to 1. The default values are all 0.

IHSTAG	0 to solve the energy equation. The dimensioning parameter NEQP must be equal to 5. (See Section 6.2.) 1 to eliminate the energy equation by assuming constant stagnation enthalpy per unit mass. This significantly lowers the overall execution time. The default value is 0.
ILAMV	0 for constant laminar viscosity and thermal conductivity coefficients equal to MUR and KTR. 1 for variable laminar viscosity and thermal conductivity coefficients, computed as a function of local temperature using Sutherland's formula for air (White, 1974). The default value is 0.

ICVARS — Parameter specifying which variables are being supplied as initial conditions for the time marching procedure by subroutine INIT. Remember that the initial conditions must be nondimensionalized by the reference conditions listed in Table 3-2. (See Section 5.0 for details on defining initial conditions.) When the energy equation is being solved (IHSTAG = 0), the allowed values are:

ICVARS	Variables Supplied By INIT
1	ρ, ρu, ρv, ρw, E_T
2	p, u, v, w, T
3	ρ, u, v, w, T
4	p, u, v, w, ρ
5	c_p, u, v, w, T
6	p, M, α_v, α_w, T

When constant stagnation enthalpy is assumed (IHSTAG = 1), the allowed values are:

ICVARS	Variables Supplied By INIT
1	ρ, ρu, ρv, ρw
2	p, u, v, w
3	ρ, u, v, w
5	c_p, u, v, w
6	p, M, α_v, α_w

In the above tables, c_p, α_v, and α_w represent static pressure coefficient, flow angle in degrees in the x-y plane, and flow angle in degrees in the x-z plane, respectively. The default value is 2.

Reference Conditions

The following parameters specify the six basic reference conditions for length, velocity, temperature, density, viscosity, and thermal conductivity. These reference conditions are used, along with some additional reference conditions derived from them, as the nondimensionalizing factors for nondimensional input and output parameters. The dimensional reference conditions may be read in using either English or SI units, depending on the value of IUNITS.

LR — Reference length L_r in feet (meters). This is a real variable. The default value is 1.0.

UR	Reference velocity u_r in ft/sec (m/sec). Either UR or MACHR may be specified, but not both. The unspecified one will be computed from the remaining reference conditions. The default value is $a_r = (\gamma_r \bar{R} T_r)^{1/2}$, the speed of sound at the reference temperature.
MACHR	Reference Mach number, $M_r = u_r/(\gamma_r \bar{R} T_r)^{1/2}$. Either MACHR or UR may be specified, but not both. The unspecified one will be computed from the remaining reference conditions. This is a real variable. The default value is 0.0.
TR	Reference temperature T_r in °R (K). The default value is 519.0 °R (288.333 K).
RHOR	Reference density ρ_r in lb_m/ft^3 (kg/m^3). The default value is 0.07645 lb_m/ft^3 (1.22461 kg/m^3).
MUR	Reference viscosity μ_r in lb_m/ft-sec (kg/m-sec). Either MUR or RER may be specified, but not both. The unspecified one will be computed from the remaining reference conditions. This is a real variable. The default value is the viscosity for air at the reference temperature TR.
RER	Reference Reynolds number, $Re_r = \rho_r u_r L_r/\mu_r$. Either RER or MUR may be specified, but not both. The unspecified one will be computed from the remaining reference conditions. The default value is 0.0.
KTR	Reference thermal conductivity k_r in lb_m-ft/sec^3-°R (kg-m/sec^3-K). Either KTR or PRLR may be specified, but not both. The unspecified one will be computed from the remaining reference conditions. This is a real variable. The default value is the thermal conductivity for air at the reference temperature TR.
PRLR	Reference laminar Prandtl number $Pr_{l_r} = c_{p_r}\mu_r/k_r$. Here c_{p_r} is the specific heat at constant pressure, defined as either $c_p(T_r)$ or $\gamma_r \bar{R}/(\gamma_r - 1)$, depending on whether or not a value is being specified for the input parameter GAMR. Either PRLR or KTR may be specified, but not both. The unspecified one will be computed from the remaining reference conditions. The default value is 0.0.

Fluid Properties

The following parameters provide information about the fluid being used.

RG	Gas constant $\bar{R}$ in ft^2/sec^2-°R (m^2/sec^2-K). The default value is 1716 ft^2/sec^2-°R (286.96 m^2/sec^2-K).
GAMR	Reference ratio of specific heats, $\gamma_r = c_{p_r}/c_{v_r}$. This parameter acts as a flag for a constant specific heat option. If a non-zero value for GAMR is specified by the user, the specific heats c_p and c_v are computed from GAMR and RG, and treated as constants. Otherwise they are computed locally as a function of temperature. The default value is 0.0.
HSTAGR	Stagnation enthalpy h_T in ft^2/sec^2 (m^2/sec^2). This parameter is only used with the constant stagnation enthalpy option (IHSTAG = 1). The default value is computed from the reference conditions.

3.1.7 Namelist BC

The parameters in this namelist specify the boundary conditions to be used for the mean flow equations and, if necessary, for the k-ε turbulence model equations. For the mean flow, NEQ conditions must be specified at each computational boundary, where NEQ is the number of coupled equations being solved. NEQ will be equal to 4 or 5 depending on the value of IHSTAG. (See Table 3-4.)

Note that the boundary conditions may be thought of as simply NEQ additional equations to be solved on the boundary. They do not necessarily have to be associated one-to-one with the governing differential equations or the dependent variables. They must, however, be functions of the dependent variables and sufficiently complete to set constraints on each of the dependent variables through their functional form. They must also, of course, be independent of one another and physically appropriate for the problem being solved.

Three different methods are available for setting boundary conditions for steady flow computations. The first, and easiest, way is to specify the type of boundary (i.e., solid wall, symmetry, etc.) using the KBC input parameters. These parameters act as "meta" flags, triggering the automatic setting of the necessary NEQ individual boundary conditions at the specified boundary.

Second, if more flexibility is needed, the NEQ individual boundary conditions may be set for each boundary using the JBC and GBC input parameters. The boundary condition *type* (specified value, specified gradient, etc.) is given by JBC, and the boundary condition *value* by GBC. With these parameters, the same conditions are applied over the entire surface.

And third, if even greater flexibility is needed, the NEQ individual boundary conditions may be set for each boundary using the IBC and FBC input parameters. These are analogous to the JBC and GBC parameters (i.e., the boundary condition type is given by IBC, and the value by FBC), but they allow a point-by-point distribution of type and value to be specified instead of using the same type and value over the entire surface.[7]

For a given boundary, boundary conditions specified via the KBC parameters override those specified using the JBC and GBC parameters, which in turn override those specified using the IBC and FBC parameters. However, the different methods may be used in combination as long as they don't conflict. For example, the KBC parameters may be used for four boundaries, the JBC and GBC parameters for the fifth boundary, and the IBC and FBC parameters for the sixth boundary. And, on a single boundary, the JBC and GBC parameters may be used for some of the NEQ boundary conditions, and the IBC and FBC parameters for the rest.

Unsteady boundary conditions may be used when individual boundary conditions are specified for the entire surface, but not when boundary conditions are specified point-by-poin..

With one exception, the NEQ boundary conditions at each boundary may be specified in any order. The exception is any condition on one of the dependent conservation variables **Q**. These must be specified in the order given in Table 3-4.

If a problem requires a boundary condition of the form $\Delta F = 0$, $F = f$, $\partial F / \partial \phi = f$, or $\nabla F \cdot \vec{n} = f$, where F is not one of the functions already built into *Proteus*, the subroutines BCF and BCFLIN may be used. This requires that the user supply subroutine BCFLIN. Subroutines BCF and BCFLIN are described in detail in Volume 3.

If the Chien k-ε turbulence model is being used, boundary conditions for k and ε must be specified in addition to those specified for the mean flow equations. (Unless a spatially periodic boundary condition is being used for the mean flow. In this case, separate boundary conditions for k and ε are not needed.) Surface boundary conditions may be specified using the JBCT/GBCT parameters, or point-by-point boundary conditions may be specified using the IBCT/FBCT parameters. These parameters are analogous to the JBC/GBC and IBC/FBC parameters used for the mean flow boundary conditions. There are no input parameters for k-ε boundary conditions that are analogous to the KBC parameters.

[7] However, note that a specified point-by-point distribution of a function value is most easily set using the "no change from initial conditions" option with the JBC parameters.

Mean Flow Boundary Types with KBC

The following parameters set mean flow boundary conditions by specifying the type of boundary (i.e., solid wall, symmetry, etc.). These parameters act as "meta" flags, triggering the automatic setting of the necessary JBC and GBC values.

KBC1	An array, given as KBC1(IBOUND), specifying the types of boundaries in the ξ direction. The subscript IBOUND = 1 or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. The default values are both 0.
KBC2	An array, given as KBC2(IBOUND), specifying the types of boundaries in the η direction. The subscript IBOUND = 1 or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. The default values are both 0.
KBC3	An array, given as KBC3(IBOUND), specifying the types of boundaries in the ζ direction. The subscript IBOUND = 1 or 2, corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. The default values are both 0.

The boundary types that may be specified are described briefly in the following table, and in greater detail in Table 3-5. For boundary types involving gradient boundary conditions, 2-point differencing is used if the input KBC value is positive, and 3-point differencing is used if it is negative. For boundary types involving "no change from initial conditions"-type boundary conditions (e.g., $\Delta T = 0$), the proper boundary values must be set in the initial conditions. When the KBC parameters are used, the corresponding GBC and FBC parameters should be defaulted.

KBC Value	Boundary Type
± 1	No-slip adiabatic wall.
± 2	No-slip wall, specified temperature.
± 3	Inviscid wall.
10	Subsonic inflow, linear extrapolation.
± 11	Subsonic inflow, zero gradient.
20	Subsonic outflow, linear extrapolation.
± 21	Subsonic outflow, zero gradient.
30	Supersonic inflow.
40	Supersonic outflow, linear extrapolation.
± 41	Supersonic outflow, zero gradient.
± 50	Symmetry.
60	Spatially periodic.

Boundary conditions specified using the KBC parameter for a given boundary override any boundary condition types specified for that boundary using the JBC or IBC parameters. Note, however, that since the default values for the KBC parameters are all 0, the default procedure for specifying boundary conditions is by using the JBC and GBC parameters.

Surface Mean Flow Boundary Condition Types and Values with JBC and GBC

The following parameters set the NEQ individual mean flow boundary condition types and values for each boundary using the JBC and GBC parameters. With these parameters, the same conditions are applied over the entire surface. Remember that the boundary condition values must be nondimensionalized by the reference conditions listed in Table 3-2. If some of the boundary conditions are being specified using the IBC and FBC parameters, the appropriate JBC parameters must be set equal to -1, as described below.

JBC1	A two-dimensional array, given as JBC1(IEQ,IBOUND), specifying the type of boundary conditions to be used on the $\xi = 0$ and $\xi = 1$ boundaries. Here IEQ = 1 to NEQ corresponding to each equation, and IBOUND = 1 or 2 corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. Setting JBC1 = -1 signals the code to use boundary conditions specified point-by-point, as given by

the input arrays IBC1 and FBC1. See Table 3-6 for a list of allowed boundary condition types. The default values are all 0.

JBC2	A two-dimensional array, given as JBC2(IEQ,IBOUND), specifying the type of boundary conditions to be used on the $\eta = 0$ and $\eta = 1$ boundaries. Here IEQ = 1 to NEQ corresponding to each equation, and IBOUND = 1 or 2 corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. Setting JBC2 = − 1 signals the code to use boundary conditions specified point-by-point, as given by the input arrays IBC2 and FBC2. See Table 3-6 for a list of allowed boundary condition types. The default values are all 0.
JBC3	A two-dimensional array, given as JBC3(IEQ,IBOUND), specifying the type of boundary conditions to be used on the $\zeta = 0$ and $\zeta = 1$ boundaries. Here IEQ = 1 to NEQ corresponding to each equation, and IBOUND = 1 or 2 corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. Setting JBC3 = − 1 signals the code to use boundary conditions specified point-by-point, as given by the input arrays IBC3 and FBC3. See Table 3-6 for a list of allowed boundary condition types. The default values are all 0.
GBC1	A two-dimensional array, given as GBC1(IEQ,IBOUND), specifying the values for the steady boundary conditions to be used on the $\xi = 0$ and $\xi = 1$ boundaries. Here IEQ = 1 to NEQ corresponding to each equation, and IBOUND = 1 or 2 corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. The default values are all 0.0.
GBC2	A two-dimensional array, given as GBC2(IEQ,IBOUND), specifying the values for the steady boundary conditions to be used on the $\eta = 0$ and $\eta = 1$ boundaries. Here IEQ = 1 to NEQ corresponding to each equation, and IBOUND = 1 or 2 corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. The default values are all 0.0.
GBC3	A two-dimensional array, given as GBC3(IEQ,IBOUND), specifying the values for the steady boundary conditions to be used on the $\zeta = 0$ and $\zeta = 1$ boundaries. Here IEQ = 1 to NEQ corresponding to each equation, and IBOUND = 1 or 2 corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. The default values are all 0.0.

Note that boundary condition types 2, 12, 22, etc., are specified values of the derivative with respect to the computational coordinate, not with respect to the physical distance in the direction of the computational coordinate. See Section 6.3 of Volume 1 for details.

Note also that normal derivative boundary condition values are positive in the direction of increasing ξ, η, or ζ. Thus, a positive value for GBC at $\xi = 0$, $\eta = 0$, or $\zeta = 0$ implies a flux into the computational domain, and a positive GBC at $\xi = 1$, $\eta = 1$, or $\zeta = 1$ implies a flux out of the computational domain. See Section 6.4 of Volume 1 for details. Similarly, the normal velocity V_n is positive in the direction of increasing ξ, η, or ζ. Thus, a positive V_n at $\xi = 0$, $\eta = 0$, or $\zeta = 0$ implies flow into the computational domain, and a positive V_n at $\xi = 1$, $\eta = 1$, or $\zeta = 1$ implies flow out of the computational domain. See the description of subroutine BCVDIR in Section 4.2 of Volume 3 for details.

Boundary conditions specified using the JBC and GBC parameters for given values of IEQ and IBOUND override any boundary conditions specified for those values of IEQ and IBOUND using the IBC and FBC parameters. Note that since the default values for the JBC parameters are all 0, the default boundary conditions are "no change from initial conditions" for the conservation variables.

Point-by-Point Mean Flow Boundary Condition Types and Values with IBC and FBC

The following parameters set the NEQ individual mean flow boundary condition types and values for each boundary using the IBC and FBC parameters. With these parameters, point-by-point distributions are specified on the surface for the boundary condition types and values. Remember that the boundary condition values must be nondimensionalized by the reference conditions listed in Table 3-2. Note that these parameters are activated by setting the appropriate JBC parameters equal to -1, as described below.

IBC1	A four-dimensional array, given as IBC1(I2,I3,IEQ,IBOUND), specifying the type of boundary conditions to be used on the $\xi = 0$ and $\xi = 1$ boundaries. Here I2 = 1 to N2 and I3 = 1 to N3 corresponding to each grid point on the boundary, IEQ = 1 to NEQ corresponding to each equation, and IBOUND = 1 or 2 corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. JBC1(IEQ,IBOUND) must be set equal to -1. See Table 3-6 for a list of allowed boundary condition types. The default values are all 0.
IBC2	A four-dimensional array, given as IBC2(I1,I3,IEQ,IBOUND), specifying the type of boundary conditions to be used on the $\eta = 0$ and $\eta = 1$ boundaries. Here I1 = 1 to N1 and I3 = 1 to N3 corresponding to each grid point on the boundary, IEQ = 1 to NEQ corresponding to each equation, and IBOUND = 1 or 2 corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. JBC2(IEQ,IBOUND) must be set equal to -1. See Table 3-6 for a list of allowed boundary condition types. The default values are all 0.
IBC3	A four-dimensional array, given as IBC3(I1,I2,IEQ,IBOUND), specifying the type of boundary conditions to be used on the $\zeta = 0$ and $\zeta = 1$ boundaries. Here I1 = 1 to N1 and I2 = 1 to N2 corresponding to each grid point on the boundary, IEQ = 1 to NEQ corresponding to each equation, and IBOUND = 1 or 2 corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. JBC3(IEQ,IBOUND) must be set equal to -1. See Table 3-6 for a list of allowed boundary condition types. The default values are all 0.
FBC1	A four-dimensional array, given as FBC1(I2,I3,IEQ,IBOUND), specifying the values for the steady boundary conditions to be used on the $\xi = 0$ and $\xi = 1$ boundaries. Here I2 = 1 to N2 and I3 = 1 to N3 corresponding to each grid point on the boundary, IEQ = 1 to NEQ corresponding to each equation, and IBOUND = 1 or 2 corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. The default values are all 0.0.
FBC2	A four-dimensional array, given as FBC2(I1,I3,IEQ,IBOUND), specifying the values for the steady boundary conditions to be used n the $\eta = 0$ and $\eta = 1$ boundaries. Here I1 = 1 to N1 and I3 = 1 to N3 corresponding to each grid point on the boundary, IEQ = 1 to NEQ corresponding to each equation, and IBOUND = 1 or 2 corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. The default values are all 0.0.
FBC3	A four-dimensional array, given as FBC3(I1,I2,IEQ,IBOUND), specifying the values for the steady boundary conditions to be used on the $\zeta = 0$ and $\zeta = 1$ boundaries. Here I1 = 1 to N1 and I2 = 1 to N2 corresponding to each grid point on the boundary, IEQ = 1 to NEQ corresponding to each equation, and IBOUND = 1 or 2 corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. The default values are all 0.0.

Note that boundary condition types 2, 12, 22, etc., are specified values of the derivative with respect to the computational coordinate, not with respect to the physical distance in the direction of the computational coordinate. See Section 6.3 of Volume 1 for details.

Note also that normal derivative boundary condition values are positive in the direction of increasing ξ, η, or ζ. Thus, a positive value for FBC at $\xi = 0$, $\eta = 0$, or $\zeta = 0$ implies a flux into the computational domain, and a positive FBC at $\xi = 1$, $\eta = 1$, or $\zeta = 1$ implies a flux out of the computational domain. See Section 6.4 of Volume 1 for details. Similarly, the normal velocity V_n is positive in the direction of increasing ξ, η, or ζ. Thus, a positive V_n at $\xi = 0$, $\eta = 0$, or $\zeta = 0$ implies flow into the computational domain, and a positive V_n at $\xi = 1$, $\eta = 1$, or $\zeta = 1$ implies flow out of the computational domain. See the description of subroutine BCVDIR in Section 4.2 of Volume 3 for details.

Unsteady Mean Flow Boundary Conditions

The following parameters are used to specify unsteady mean flow boundary conditions. The boundary condition type (specified value, specified gradient, etc.) is given by JBC, as described above, but the value is given by GTBC. The type of unsteadiness (general or periodic) is given by JTBC.

JTBC1

A two-dimensional array, given as JTBC1(IEQ,IBOUND), specifying the type of time dependency for the boundary conditions on the $\xi = 0$ and $\xi = 1$ boundaries. Here IEQ = 1 to NEQ corresponding to each equation, and IBOUND = 1 or 2 corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. Valid values of JTBC1(IEQ,IBOUND) are:

0 for a steady boundary condition, whose value is given by GBC1.
1 for a general unsteady boundary condition, whose value is determined by linear interpolation in the input table of GTBC1 vs. NTBCA.
2 for a time-periodic boundary condition of the form $g_1 + g_2 \sin(g_3 n + g_4)$, where n is the time level and g_1 through g_4 are given by the first four values of GTBC1.

The default values are all 0.

JTBC2

A two-dimensional array, given as JTBC2(IEQ,IBOUND), specifying the type of time dependency for the boundary conditions on the $\eta = 0$ and $\eta = 1$ boundaries. Here IEQ = 1 to NEQ corresponding to each equation, and IBOUND = 1 or 2 corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. Valid values of JTBC2(IEQ,IBOUND) are:

0 for a steady boundary condition, whose value is given by GBC2.
1 for a general unsteady boundary condition, whose value is determined by linear interpolation in the input table of GTBC2 vs. NTBCA.
2 for a time-periodic boundary condition of the form $g_1 + g_2 \sin(g_3 n + g_4)$, where n is the time level and g_1 through g_4 are given by the first four values of GTBC2.

The default values are all 0.

JTBC3

A two-dimensional array, given as JTBC3(IEQ,IBOUND), specifying the type of time dependency for the boundary conditions on th. $\zeta = 0$ and $\zeta = 1$ boundaries. Here IEQ = 1 to NEQ corresponding to each equation, and IBOUND = 1 or 2 corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. Valid values of JTBC3(IEQ,IBOUND) are:

0 for a steady boundary condition, whose value is given by GBC3.
1 for a general unsteady boundary condition, whose value is determined by linear interpolation in the input table of GTBC3 vs. NTBCA.
2 for a time-periodic boundary condition of the form $g_1 + g_2 \sin(g_3 n + g_4)$, where n is the time level and g_1 through g_4 are given by the first four values of GTBC3.

The default values are all 0.

NTBC

Number of values in the tables of GTBC1, GTBC2, and/or GTBC3 vs. NTBCA for the general unsteady boundary condition option. The maximum value allowed is the value of the dimensioning parameter NTP. (See Section 6.2.) The default value is 0.

NTBCA

An array of NTBC time levels at which GTBC1, GTBC2, and/or GTBC3 are specified for the general unsteady boundary condition option. The default values are all 0.

GTBC1

A three-dimensional array, given as GTBC1(ITBC,IEQ,IBOUND), used in the unsteady and time-periodic boundary condition options for the $\xi = 0$ and $\xi = 1$

boundaries. Here IEQ = 1 to NEQ corresponding to each equation, and IBOUND = 1 or 2 corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. For general unsteady boundary conditions the subscript ITBC = 1 to NTBC, corresponding to the time levels in the array NTBCA, and GTBC1 specifies the boundary condition value directly. For time-periodic boundary conditions the subscript ITBC = 1 to 4, and GTBC1 specifies the four coefficients in the equation used to determine the boundary condition value. The default values are all 0.0.

GTBC2 — A three-dimensional array, given as GTBC2(ITBC,IEQ,IBOUND), used in the unsteady and time-periodic boundary condition options for the $\eta = 0$ and $\eta = 1$ boundaries. Here IEQ = 1 to NEQ corresponding to each equation, and IBOUND = 1 or 2 corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. For general unsteady boundary conditions the subscript ITBC = 1 to NTBC, corresponding to the time levels in the array NTBCA, and GTBC2 specifies the boundary condition value directly. For time-periodic boundary conditions the subscript ITBC = 1 to 4, and GTBC2 specifies the four coefficients in the equation used to determine the boundary condition value. The default values are all 0.0.

GTBC3 — A three-dimensional array, given as GTBC3(ITBC,IEQ,IBOUND), used in the unsteady and time-periodic boundary condition options for the $\zeta = 0$ and $\zeta = 1$ boundaries. Here IEQ = 1 to NEQ corresponding to each equation, and IBOUND = 1 or 2 corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. For general unsteady boundary conditions the subscript ITBC = 1 to NTBC, corresponding to the time levels in the array NTBCA, and GTBC3 specifies the boundary condition value directly. For time-periodic boundary conditions the subscript ITBC = 1 to 4, and GTBC3 specifies the four coefficients in the equation used to determine the boundary condition value. The default values are all 0.0.

k-ε Surface Boundary Condition Types and Values with JBCT and GBCT

The following parameters set the individual k and ε boundary condition types and values for each boundary using the JBCT and GBCT parameters. With these parameters, the same conditions are applied over the entire boundary. Remember that the boundary condition values must be nondimensionalized by the reference conditions listed in Table 3-2. None of the following parameters are needed if spatially periodic boundary conditions are being used. If either of the boundary conditions for a computational boundary is being specified using the IBCT and FBCT parameters, the appropriate JBCT parameters must be set equal to -1, as described below.

JBCT1 — A two-dimensional array, given as JBCT1(IEQ,IBOUND), specifying the type of boundary conditions to be used on the $\xi = 0$ and $\xi = 1$ boundaries. Here IEQ = 1 or 2 corresponding to k and ε, respectively, and IBOUND = 1 or 2 corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. Setting JBCT1 = -1 signals the code to use boundary conditions specified point-by-point, as given by the input arrays IBCT1 and FBCT1. See Table 3-7 for a list of allowed boundary condition types. The default values are 0 for IEQ = 1, and 10 for IEQ = 2.

JBCT2 — A two-dimensional array, given as JBCT2(IEQ,IBOUND), specifying the type of boundary conditions to be used on the $\eta = 0$ and $\eta = 1$ boundaries. Here IEQ = 1 or 2 corresponding to k and ε, respectively, and IBOUND = 1 or 2 corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. Setting JBCT2 = -1 signals the code to use boundary conditions specified point-by-point, as given by the input arrays IBCT2 and FBCT2. See Table 3-7 for a list of allowed boundary condition types. The default values are 0 for IEQ = 1, and 10 for IEQ = 2.

JBCT3 — A two-dimensional array, given as JBCT3(IEQ,IBOUND), specifying the type of boundary conditions to be used on the $\zeta = 0$ and $\zeta = 1$ boundaries. Here IEQ = 1 or 2 corresponding to k and ε, respectively, and IBOUND = 1 or 2 corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. Setting JBCT3 = -1 signals the code to use boundary conditions specified point-by-point, as given by the input arrays IBCT3 and FBCT3. See Table 3-7 for a list of allowed boundary condition types. The default values are 0 for IEQ = 1, and 10 for IEQ = 2.

GBCT1	A two-dimensional array, given as GBCT1(IEQ,IBOUND), specifying the values for the boundary conditions to be used on the $\xi = 0$ and $\xi = 1$ boundaries. Here IEQ = 1 or 2 corresponding to k and ε, respectively, and IBOUND = 1 or 2 corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. The default values are all 0.0.
GBCT2	A two-dimensional array, given as GBCT2(IEQ,IBOUND), specifying the values for the boundary conditions to be used on the $\eta = 0$ and $\eta = 1$ boundaries. Here IEQ = 1 or 2 corresponding to k and ε, respectively, and IBOUND = 1 or 2 corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. The default values are all 0.0.
GBCT3	A two-dimensional array, given as GBCT3(IEQ,IBOUND), specifying the values for the boundary conditions to be used on the $\zeta = 0$ and $\zeta = 1$ boundaries. Here IEQ = 1 or 2 corresponding to k and ε, respectively, and IBOUND = 1 or 2 corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. The default values are all 0.0.

Note that boundary condition types 2 and 12 are specified values of the derivative with respect to the computational coordinate, not with respect to the physical distance in the direction of the computational coordinate. See Section 6.3 of Volume 1 for details.

Boundary conditions specified using the JBCT and GBCT parameters for given values of IEQ and IBOUND will override any boundary conditions specified for those values of IEQ and IBOUND using the IBCT and FBCT parameters. Note that the default values for the JBCT parameters are "no change from initial conditions" for the k and ε.

k-ε Point-by-Point Boundary Condition Types and Values with IBCT and FBCT

The following parameters set the individual k and ε boundary condition types and values for each boundary using the IBCT and FBCT parameters. With these parameters, point-by-point distributions are specified on the boundary for the boundary condition types and values. Remember that the boundary condition values must be nondimensionalized by the reference conditions list. 1 in Table 3-2. None of the following parameters are needed if spatially periodic boundary conditions are being used. Note that these parameters are activated by setting the appropriate JBCT parameters equal to -1, as described below.

IBCT1	A four-dimensional array, given as IBCT1(I2,I3,IEQ,IBOUND), specifying the type of boundary conditions to be used on the $\xi = 0$ and $\xi = 1$ boundaries. Here I2 = 1 to N2 and I3 = 1 to N3 corresponding to each grid point on the boundary, IEQ = 1 or 2 corresponding to k and ε, respectively, and IBOUND = 1 or 2 corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. JBCT1(IEQ,IBOUND) must be set equal to -1. See Table 3-7 for a list of allowed boundary condition types. The default values are 0 for IEQ = 1, and 10 for IEQ = 2.
IBCT2	A four-dimensional array, given as IBCT2(I1,I3,IEQ,IBOUND), specifying the type of boundary conditions to be used on the $\eta = 0$ and $\eta = 1$ boundaries. Here I1 = 1 to N1 and I3 = 1 to N3 corresponding to each grid point on the boundary, IEQ = 1 or 2 corresponding to k and ε, respectively, and IBOUND = 1 or 2 corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. JBCT2(IEQ,IBOUND) must be set equal to -1. See Table 3-7 for a list of allowed boundary condition types. The default values are 0 for IEQ = 1, and 10 for IEQ = 2.
IBCT3	A four-dimensional array, given as IBCT3(I1,I2,IEQ,IBOUND), specifying the type of boundary conditions to be used on the $\zeta = 0$ and $\zeta = 1$ boundaries. Here

I1 = 1 to N1 and I2 = 1 to N2 corresponding to each grid point on the boundary, IEQ = 1 or 2 corresponding to k and ε, respectively, and IBOUND = 1 or 2 corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. JBCT3(IEQ,IBOUND) must be set equal to -1. See Table 3-7 for a list of allowed boundary condition types. The default values are 0 for IEQ = 1, and 10 for IEQ = 2.

FBCT1	A four-dimensional array, given as FBCT1(I2,I3,IEQ,IBOUND), specifying the values for the boundary conditions to be used on the $\xi = 0$ and $\xi = 1$ boundaries. Here I2 = 1 to N2 and I3 = 1 to N3 corresponding to each grid point on the boundary, IEQ = 1 or 2 corresponding to k and ε, respectively, and IBOUND = 1 or 2 corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. The default values are all 0.0.
FBCT2	A four-dimensional array, given as FBCT2(I1,I3,IEQ,IBOUND), specifying the values for the boundary conditions to be used on the $\eta = 0$ and $\eta = 1$ boundaries. Here I1 = 1 to N1 and I3 = 1 to N3 corresponding to each grid point on the boundary, IEQ = 1 or 2 corresponding to k and ε, respectively, and IBOUND = 1 or 2 corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. The default values are all 0.0.
FBCT3	A four-dimensional array, given as FBCT3(I1,I2,IEQ,IBOUND), specifying the values for the boundary conditions to be used on the $\zeta = 0$ and $\zeta = 1$ boundaries. Here I1 = 1 to N1 and I2 = 1 to N2 corresponding to each grid point on the boundary, IEQ = 1 or 2 corresponding to k and ε, respectively, and IBOUND = 1 or 2 corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. The default values are all 0.0.

Note that boundary condition types 2 and 12 are specified values of the derivative with respect to the computational coordinate, not with respect to the physical distance in the direction of the computational coordinate. See Section 6.3 of Volume 1 for details.

3.1.8 Namelist NUM

Mesh Parameters

The following parameters specify the number of mesh points and the degree of packing.

N1	Number of grid points N_1 in the ξ direction. For non-periodic boundary conditions in the ξ direction, the maximum value allowed is the value of the dimensioning parameter N1P. For spatially periodic boundary conditions, the maximum is N1P $-$ 1. (See Section 6.2.) The default value is 5.
N2	Number of grid points N_2 in the η direction. For non-periodic boundary conditions in the η direction, the maximum value allowed is the value of the dimensioning parameter N2P. For spatially periodic boundary conditions, the maximum is N2P $-$ 1. (See Section 6.2.) The default value is 5.
N3	Number of grid points N_3 in the ζ direction. For non-periodic boundary conditions in the ζ direction, the maximum value allowed is the value of the dimensioning parameter N3P. For spatially periodic boundary conditions, the maximum is N3P $-$ 1. (See Section 6.2.) The default value is 5.
IPACK	A 3-element array, specified as IPACK(IDIR), indicating whether or not grid points are to be packed in direction IDIR. The subscript IDIR = 1, 2, or 3, corresponding to the ξ, η, and ζ directions, respectively. Valid values of IPACK(IDIR) are:

0 for no packing in direction IDIR.

1 to pack points in direction IDIR using a transformation due to Roberts (1971). The location and amount of packing are specified by the array SQ.

The default values are all 0.

SQ

A two-dimensional array controlling the packing of grid points near computational boundaries, specified as SQ(IDIR,IPC). The subscript IDIR = 1, 2, or 3 corresponding to packing in the ξ, η, and ζ directions, respectively. The subscript IPC = 1 or 2, where SQ(IDIR,1) specifies the packing location, and SQ(IDIR,2) specifies the amount of packing.

Meaningful values for SQ(IDIR,1) are 0.0, 0.5, and 1.0, where 0.0 corresponds to packing near the lower boundary only (i.e., at ξ, η, or $\zeta = 0$, depending on IDIR), 1.0 corresponds to packing near the upper boundary only, and 0.5 corresponds to equal packing at both boundaries.

Meaningful values for SQ(IDIR,2) are values above 1.0, but generally 1.1 or below. The closer SQ(IDIR,2) is to 1.0, the tighter the packing will be.

The default values are SQ(IDIR,1) = 0.0 and SQ(IDIR,2) = 1.01 for IDIR = 1, 2, and 3.

Artificial Viscosity Parameters

The following parameters specify the type and amount of artificial viscosity to be used.

IAV4E

0 for no fourth-order explicit artificial viscosity.
1 to include fourth-order explicit artificial viscosity using the constant coefficient model of Steger (1978).
2 to include fourth-order explicit artificial viscosity using the nonlinear coefficient model of Jameson, Schmidt, and Turkel (1981).

The default value is 1.

IAV2E

0 for no second-order explicit artificial viscosity.
1 to include second-order explicit artificial viscosity using the constant coefficient model.
2 to include second-order explicit artificial viscosity using the nonlinear coefficient model of Jameson, Schmidt, and Turkel (1981).

The default value is 0.

IAV2I

0 for no second-order implicit artificial viscosity.
1 to include second-order implicit artificial viscosity using the constant coefficient model of Steger (1978).

The default value is 1.

CAVS4E

For the constant coefficient model, CAVS4E(IEQ) specifies the fourth-order artificial viscosity coefficient $\varepsilon_e^{(4)}$ directly. For the nonlinear coefficient model it specifies the constant κ_4. The subscript IEQ varies from 1 to NEQ corresponding to each coupled equation. (See Table 3-4 for the order of the equations being solved.) Good values for a given application are usually determined by experience, but recommended starting values are 1.0 for the constant coefficient model, 0.005 for the nonlinear model when spatially varying second-order time differencing is used, and 0.0002 for the nonlinear model when a spatially constant first-order time differencing is used. The default values are all 1.0.

CAVS2E	For the constant coefficient model, CAVS2E(IEQ) specifies the second-order artificial viscosity coefficient $\varepsilon_E^{(2)}$ directly. For the nonlinear coefficient model it specifies the constant κ_2. The subscript IEQ varies from 1 to NEQ corresponding to each coupled equation. (See Table 3-4 for the order of the equations being solved.) Good values for a given application are usually determined by experience, but recommended starting values are 1.0 for the constant coefficient model, 0.01 for the nonlinear model for flows without shocks, and 0.1 for the nonlinear model for flows with shocks. The default values are all 1.0.
CAVS2I	Second-order implicit artificial viscosity coefficient, ε_I, specified as CAVS2I(IEQ). The subscript IEQ varies from 1 to NEQ corresponding to each coupled equation. (See Table 3-4 for the order of the equations being solved.) Good values for a given application are usually determined by experience, but recommended starting values are 2.0 for the constant coefficient model, and 0.0 for the nonlinear model. The default values are all 2.0.

Time Difference Centering Parameters

The following parameters specify the type of time differencing scheme to be used. The generalized Beam and Warming (1978) time differencing formula is given by equation (3.1) of Volume 1.

THC	A 2-element array specifying the time differencing centering parameters θ_1 and θ_2 to be used for the continuity equation. The default values are 1.0 and 0.0.
THX	A 3-element array specifying the time differencing centering parameters θ_1, θ_2, and θ_3 to be used for the x-momentum equation. The default values are 1.0, 0.0, and 0.0.
THY	A 3-element array specifying the time differencing centering parameters θ_1, θ_2, and θ_3 to be used for the y-momentum equation. The default values are 1.0, 0.0, and 0.0.
THZ	A 3-element array specifying the time differencing centering parameters θ_1, θ_2, and θ_3 to be used for the z-momentum equation. The default values are 1.0, 0.0, and 0.0.
THE	A 3-element array specifying the time differencing centering parameters θ_1, θ_2, and θ_3 to be used for the energy equation. The default values are 1.0, 0.0, and 0.0.
THKE	A 2-element array specifying the time differencing centering parameters θ_1 and θ_2 to be used for the k-ε equations. The default values are 1.0 and 0.0.

The following table summarizes the time differencing schemes that may be used. The Euler implicit method is recommended for steady flows, and the 3-point backward implicit method is recommended for unsteady flows.

θ_1	θ_2	θ_3	Method	Accuracy
1	0	0	Euler implicit	$O(\Delta\tau)$
1/2	0	1/2	Trapezoidal implicit	$O(\Delta\tau)^2$
1	1/2	1	3-point backward implicit	$O(\Delta\tau)^2$

3.1.9 Namelist TIME

Time Step Selection Parameters

The following parameters determine the procedure used to set the time step size for the mean flow equations, and to change it as the solution proceeds. The various options for IDTAU are described in more detail in the description of subroutine TIMSTP in Section 4.2 of Volume 3.

IDTMOD	The time step size $\Delta\tau$ is recomputed every IDTMOD'th step. The default value is 100000.
IDTAU	1 for a *global* (i.e., constant in space) time step $\Delta\tau = (\mathrm{CFL})\Delta\tau_{cfl}$, where $\Delta\tau_{cfl}$ is the minimum of the allowable time steps at each grid point based on the CFL criteria for explicit methods. 2 for a global time step initially computed using the IDTAU = 1 option, but adjusted as the solution proceeds based on $\Delta\mathbf{Q}_{max}$, the absolute value of the maximum change in the dependent variables.[8] For any of the dependent variables, if $\Delta\mathbf{Q}_{max} < \mathrm{CHG1}$, the CFL number is multiplied by DTF1. If $\Delta\mathbf{Q}_{max} > \mathrm{CHG2}$, the CFL number is divided by DTF2. If $\Delta\mathbf{Q}_{max} > 0.15$, the CFL number is cut in half. The CFL number will not be decreased below CFLMIN, or increased above CFLMAX. 3 for a global time step $\Delta\tau$ equal to the specified input DT. 4 for a global time step initially equal to the specified input DT, but adjusted as the solution proceeds based on $\Delta\mathbf{Q}_{max}$, the absolute value of the maximum change in the dependent variables.[8] For any of the dependent variables, if $\Delta\mathbf{Q}_{max} < \mathrm{CHG1}$, $\Delta\tau$ is multiplied by DTF1. If $\Delta\mathbf{Q}_{max} > \mathrm{CHG2}$, $\Delta\tau$ is divided by DTF2. If $\Delta\mathbf{Q}_{max} > 0.15$, $\Delta\tau$ is cut in half. $\Delta\tau$ will not be decreased below DTMIN, or increased above DTMAX. 5 for a *local* (i.e., varying in space) time step $\Delta\tau = (\mathrm{CFL})\Delta\tau_{cfl}$, where $\Delta\tau_{cfl}$ is the allowable time step at each grid point based on the CFL criteria for explicit methods. 6 for a local time step initially computed using the IDTAU = 5 option, but adjusted as the solution proceeds based on $\Delta\mathbf{Q}_{max}$, the absolute value of the maximum change in the dependent variables.[8] For any of the dependent variables, if $\Delta\mathbf{Q}_{max} < \mathrm{CHG1}$, the CFL number is multiplied by DTF1. If $\Delta\mathbf{Q}_{max} > \mathrm{CHG2}$, the CFL number is divided by DTF2. If $\Delta\mathbf{Q}_{max} > 0.15$, the CFL number is cut in half. The CFL number will not be decreased below CFLMIN, or increased above CFLMAX. 7 for a global time step with cycling. $\Delta\tau$ will be cycled repeatedly between DTMIN and DTMAX using a logarithmic progression over NDTCYC time steps. For some problems this option has been shown to dramatically speed convergence. However, the choice of DTMIN, DTMAX, and NDTCYC is critical, and no method has been developed that assures a good choice. Poor choices may even slow down convergence, so this option should be used with caution. 8 for a local time step computed using the procedure of Knight and Choi (1989). With this option, $\Delta\tau = \max[\Delta\tau_0, (\Delta\tau_{cfl})_\xi]$, where $\Delta\tau_0 = (\mathrm{CFL})\min[(\Delta\tau_{cfl})_\xi, (\Delta\tau_{cfl})_\eta, (\Delta\tau_{cfl})_\zeta]$. The parameters $(\Delta\tau_{cfl})_\xi$, $(\Delta\tau_{cfl})_\eta$, and $(\Delta\tau_{cfl})_\zeta$ are the allowable time steps at each grid point based on the CFL criteria for explicit methods, computed separately for each computational coordinate direction. This formulation assumes that flow is generally in the ξ direction. 9 for a local time step computed as in the IDTAU = 8 option, but with a viscous correction added to the definitions of $(\Delta\tau_{cfl})_\xi$, $(\Delta\tau_{cfl})_\eta$, and $(\Delta\tau_{cfl})_\zeta$, similar to that used by Cooper (1987).

[8] In $\Delta\mathbf{Q}_{max}$, the total energy $\overline{E}_T$ has been divided by $E_{T_r} = \rho_r \overline{R} T_r/(\gamma_r - 1) + \rho_r u_r^2/2$ so that it is the same order of magnitude as the other conservation variables.

If IDTAU = 7, ICHECK and IDTMOD are both automatically set equal to 1, and NITAVG is set equal to NDTCYC. In addition, if IDTAU = 7 and ICTEST = 1, ICTEST is changed to 2. If IDTAU = 2, 4, or 6, IDTMOD is automatically set equal to ICHECK. The default value is 5.

The above parameters IDTAU and IDTMOD apply to every case. Which of the remaining parameters are needed depends on the value of IDTAU, as specified in the following table.

IDTAU	Parameters Needed
1	CFL
2	CFL, CHG1, CHG2, DTF1, DTF2, CFLMIN, CFLMAX
3	DT
4	DT, CHG1, CHG2, DTF1, DTF2, DTMIN, DTMAX
5	CFL
6	CFL, CHG1, CHG2, DTF1, DTF2, CFLMIN, CFLMAX
7	DTMIN, DTMAX, NDTCYC
8	CFL
9	CFL

CFL — An array, given as CFL(ITSEQ), specifying the ratio $\Delta\tau/\Delta\tau_{cfl}$, where $\Delta\tau$ is the actual time step used in the implicit calculation and $\Delta\tau_{cfl}$ is the allowable time step based on the CFL criteria for explicit methods. The subscript ITSEQ is the sequence number. For time steps 1 through NTIME(1), CFL(1) will be used. Then for steps NTIME(1) + 1 through NTIME(1) + NTIME(2), CFL(2) will be used, etc.[9] CFL is only used if IDTAU = 1, 2, 5, 6, 8, or 9. The default values are all 1.0.

DT — An array, given as DT(ITSEQ), specifying the time step size $\Delta\tau$. The subscript ITSEQ is the sequence number. For time steps 1 through NTIME(1), DT(1) will be used. Then for steps NTIME(1) + 1 through NTIME(1) + NTIME(2), DT(2) will be used, etc.[10] DT is only used if IDTAU = 3 or 4. The default values are all 0.01.

CHG1 — Minimum change, in absolute value, that is allowed in any dependent variable before increasing $\Delta\tau$. CHG1 is only used if IDTAU = 2, 4, or 6. The default value is 0.04.

CHG2 — Maximum change, in absolute value, that is allowed in any dependent variable before decreasing $\Delta\tau$. CHG2 is only used if IDTAU = 2, 4, or 6. The default value is 0.06.

DTF1 — Factor by which $\Delta\tau$ is multiplied if the solution changes too slowly. DTF1 is only used if IDTAU = 2, 4, or 6. The default value is 1.25.

DTF2 — Factor by which $\Delta\tau$ is divided if the solution changes too quickly. DTF2 is only used if IDTAU = 2, 4, or 6. The default value is 1.25.

CFLMIN — Minimum value that the CFL number is allowed to reach. CFLMIN is only used if IDTAU = 2 or 6. The default value is 0.5.

[9] Note that if IDTAU = 2 or 6, CFL(1) only sets $\Delta\tau$ for the first time step, and that the time step sequencing option does not apply.

[10] Note that if IDTAU = 4, DT(1) only sets $\Delta\tau$ for the first time step, and that the time step sequencing option does not apply.

CFLMAX	Maximum value that the CFL number is allowed to reach. CFLMAX is only used if IDTAU = 2 or 6. The default value is 10.0.
DTMIN	Minimum value that $\Delta\tau$ is allowed to reach (IDTAU = 4), or the minimum $\Delta\tau$ in the time step cycling procedure (IDTAU = 7.) The default value is 0.1.
DTMAX	Maximum value that $\Delta\tau$ is allowed to reach (IDTAU = 4), or the maximum $\Delta\tau$ in the time step cycling procedure (IDTAU = 7.) The default value is 0.1.
NDTCYC	Number of time steps per time step cycle. NDTCYC is used only with IDTAU = 7. The default value is 2, which results in a constant $\Delta\tau$ if DTMIN = DTMAX.

Time Marching Limits

These parameters determine the maximum number of time steps that will be taken.

NTSEQ	The number of time step sequences being used. The maximum value allowed is the value of the dimensioning parameter NTSEQP. If NTSEQ > 1, IDTAU must be equal to 1, 3, or 5. (See Section 6.2.) The default value is 1.
NTIME	An array, given as NTIME(ITSEQ), specifying the maximum number of time steps to march. The subscript ITSEQ varies from 1 to NTSEQ, and allows a series of different time steps to be specified by the values of CFL or DT. NTIME(ITSEQ) specifies the number of time steps within sequence ITSEQ. If NTSEQ = 3, for example, the total number of time steps taken will be N_{total} = NTIME(1) + NTIME(2) + NTIME(3). The initial time level is level 1, and the final computed time level will be level $N_{total} + 1$. The default values are 10, 9*0.
TLIM	When the amount of CPU time remaining for the job drops below TLIM seconds, the calculation is stopped. The default value is 20.0.

Convergence Testing Parameters

These parameters determine the convergence criteria to be used.

ICHECK	Results are checked for convergence every ICHECK'th time level. The default value is 10.
ICTEST	1 to determine convergence based on the maximum change in absolute value of each of the conservation variables over a single time step, ΔQ_{max}.[11] 2 to determine convergence based on the maximum change in absolute value of each of the conservation variables, averaged over the last NITAVG time steps, ΔQ_{avg}.[11] 3 to determine convergence based on R_{L_2}, the L_2 norm of the residual for each equation. 4 to determine convergence based on R_{avg}, the average absolute value of the residual for each equation. 5 to determine convergence based on R_{max}, the maximum absolute value of the residual for each equation. Convergence is assumed when the maximum change or residual parameter is less than EPS. Note that the change in conservation variables over a time step is directly related to the size of the time step. Small time steps naturally yield small

[11] The total energy $\overline{E}_T$ is divided by $E_{T_r} = \rho_r \overline{R} T_r/(\gamma_r - 1) + \rho_r u_r^2/2$ before testing for convergence, so that it is the same order of magnitude as the other conservation variables.

changes in conservation variables. With ICTEST = 1 or 2, therefore, convergence may be indicated prematurely.

If ICTEST = 2, ICHECK and IDTMOD are automatically set equal to 1. The default value is 3.

EPS	Level of convergence to be reached, specified as EPS(IVAR) where IVAR varies from 1 to NEQ, corresponding to each conservation variable or equation. The default values are all 0.001.
NITAVG	Number of time steps over which the maximum change in conservation variables is averaged to determine convergence. The maximum value allowed is the value of the dimensioning parameter NAMAX. (See Section 6.2.) NITAVG only applies to the ICTEST = 2 option. The default value is 10.

3.1.10 Namelist TURB

Control Parameters

The following parameters determine the type of turbulence model that will be used, and where it will be applied. These parameters apply to both the Baldwin-Lomax algebraic model and the Chien two-equation model.

ITURB

0 for laminar flow.
1 for turbulent flow, using the algebraic eddy viscosity model of Baldwin and Lomax (1978), as described in Section 9.0 of Volume 1.
20 for turbulent flow, using the two-equation k-ε model of Chien (1982), as described in Section 9.0 of Volume 1.

The default value is 0.

PRT

If PRT > 0.0, it specifies the turbulent Prandtl number, which will be treated as constant. If PRT $\leq$ 0.0, the turbulent Prandtl number will vary, and be computed using the empirical formula of Wassel and Catton (1973). The default value is 0.91.

LWALL1

A three-dimensional array, given as LWALL1(I2,I3,IBOUND), specifying which points on the ξ boundaries are on a solid wall. Here I2 = 1 to N2 and I3 = 1 to N3 corresponding to each grid point on the boundary, and IBOUND = 1 or 2, corresponding to the $\xi = 0$ and $\xi = 1$ boundaries, respectively. Valid values of LWALL1(I2,I3,IBOUND) are:

0 if the I2,I3 point on boundary IBOUND is not on a solid wall.
1 if the I2,I3 point on boundary IBOUND is on a solid wall.

If LWALL1 is not specified, wall points on the ξ boundaries are identified by looking for points with zero velocity. Therefore, LWALL1 need only be specified for cases with non-zero velocity at the wall (e.g., a moving wall, bleed, blowing, etc.). LWALL1 is not needed if the boundary condition for boundary IBOUND is set using the KBC1(IBOUND) meta flag. The default values are all 0.

LWALL2

A three-dimensional array, given as LWALL2(I1,I3,IBOUND), specifying which points on the η boundaries are on a solid wall. Here I1 = 1 to N1 and I3 = 1 to N3 corresponding to each grid point on the boundary, and IBOUND = 1 or 2, corresponding to the $\eta = 0$ and $\eta = 1$ boundaries, respectively. Valid values of LWALL2(I1,I3,IBOUND) are:

0 if the I1,I3 point on boundary IBOUND is not on a solid wall.
1 if the I1,I3 point on boundary IBOUND is on a solid wall.

If LWALL2 is not specified, wall points on the η boundaries are identified by looking for points with zero velocity. Therefore, LWALL2 need only be specified for cases with non-zero velocity at the wall (e.g., a moving wall, bleed, blowing, etc.). LWALL2 is not needed if the boundary condition for boundary IBOUND is set using the KBC2(IBOUND) meta flag. The default values are all 0.

LWALL3

A three-dimensional array, given as LWALL3(I1,I2,IBOUND), specifying which points on the ζ boundaries are on a solid wall. Here I1 = 1 to N1 and I2 = 1 to N2 corresponding to each grid point on the boundary, and IBOUND = 1 or 2, corresponding to the $\zeta = 0$ and $\zeta = 1$ boundaries, respectively. Valid values of LWALL3(I1,I2,IBOUND) are:

0 if the I1,I2 point on boundary IBOUND is not on a solid wall.
1 if the I1,I2 point on boundary IBOUND is on a solid wall.

If LWALL3 is not specified, wall points on the ζ boundaries are identified by looking for points with zero velocity. Therefore, LWALL3 need only be specified for cases with non-zero velocity at the wall (e.g., a moving wall, bleed, blowing, etc.). LWALL3 is not needed if the boundary condition for boundary IBOUND is set using the KBC3(IBOUND) meta flag. The default values are all 0.

Baldwin-Lomax Turbulence Model Parameters

These parameters apply only to the Baldwin-Lomax algebraic eddy viscosity model. Note, however, that they are used when the Baldwin-Lomax model is used to generate initial turbulent viscosity values for the Chien k-ε model (i.e., when IREST = 0, 1, or 3.)

The following two options may be used to modify the Baldwin-Lomax model.

INNER

1 to use the inner layer model of Baldwin and Lomax (1978).
2 to use the inner layer model of Spalding (1961) and Kleinstein (1967).

The default value is 1.

ILDAMP

0 to use the normal Baldwin-Lomax mixing length formula in the inner region.
1 to use the modified mixing length formula of Lau der and Priddin (1973) in the inner region of the Baldwin-Lomax model.

The default value is 0.

The following parameters are various constants used in the Baldwin-Lomax model.

CCLAU

The Clauser constant K used in the Baldwin-Lomax outer region model. The default value is 0.0168.

CCP

The constant C_{cp} used in the Baldwin-Lomax outer region model. The default value is 1.6.

CWK

The constant C_{wk} used in the formula for F_{wake} in the Baldwin-Lomax outer region model. The default value is 0.25.

CKLEB

The constant C_{Kleb} used in the formula for the Klebanoff intermittency factor F_{Kleb} in the Baldwin-Lomax outer region model. The default value is 0.3.

CKMIN

The constant $(C_{Kleb})_{min}$ used in the formula for the Klebanoff intermittency factor F_{Kleb} in the Baldwin-Lomax outer region model. The default value is normally 0.0. However, when the Baldwin-Lomax model is being used to generate initial turbulent viscosity values for the Chien k-ε model (ITURB = 20 and IREST = 0, 1, or 3), the default value is 0.1.

APLUS	The Van Driest damping constant A^+ used in the Baldwin-Lomax outer and inner region models. The default value is 26.0.
CB	The constant B used in the formula for the Klebanoff intermittency factor F_{Kleb} in the Baldwin-Lomax outer region model, and in the Spalding-Kleinstein inner region model. The default value is 5.5.
CVK	The Von Karman mixing length constant κ used in both the Baldwin-Lomax and Spalding-Kleinstein inner region models. The default value is 0.4.
CNL	The exponent n in the Launder-Priddin modified mixing length formula. The default value is 1.7.

Chien Turbulence Model Parameters

These parameters apply only to the Chien k-ε model. The following two parameters specify the time step size for the k-ε equations.

NTKE	Number of k-ε time iterations per mean flow time iteration. Note that this must be an integer equal to or greater than 1. Past experience indicates that this value should generally be less than 50. The default value is 1.
TFACT	Factor used in computing the k-ε time step. Note that this can be any real positive number. The time step size for the k-ε equations will be equal to $\Delta\tau_{k\text{-}\varepsilon} = \text{TFACT}(\Delta\tau)$, where $\Delta\tau$ is the time step size for the mean flow equations. For equal k-ε and mean flow time steps, use TFACT = 1/NTKE. The default value is 1.0.

The following parameters are various constants used in the Chien k-ε model.

CMUR	Constant C_{μ_r} used to compute C_μ in the turbulent viscosity formula for the k-ε equations. The default value is 0.09.
CONE	Constant C_1 used in the production term of the ε equation. The default value is 1.35.
CTHREE	Constant C_3 used to compute C_μ in the turbulent viscosity formula for the k-ε equations. The default value is 0.0115.
CTWOR	Constant C_{2_r} used to compute C_2 in the dissipation term of the ε equation. The default value is 1.8.
SIGE	The constant σ_ε used in the diffusion term of the ε equation. The default value is 1.3.
SIGK	The constant σ_k used in the diffusion term of the k equation. The default value is 1.0.

3.1.11 Namelist IC

This namelist is used in subroutine INIT to read in parameters needed in setting up the initial conditions for the mean flow. The version of INIT built into *Proteus* specifies uniform flow with constant properties everywhere in the flow field. In general, however, the user will supply a version of INIT tailored to the problem being solved. This namelist, then, may be modified by the user to read in parameters different from those listed here.

P0	Initial static pressure p_0. The default value is 1.0.
T0	Initial static temperature T_0. The default value is 1.0.

U0	Initial x-direction velocity u_0. The default value is 0.0.
V0	Initial y-direction velocity v_0. The default value is 0.0.
W0	Initial z-direction velocity w_0. The default value is 0.0.

3.2 COORDINATE SYSTEM FILE

The type of computational coordinate system to be used is controlled by the input parameter NGEOM in namelist GMTRY. For NGEOM = 10, the coordinate system is read from a pre-stored file. This file may be created by any body-fitted coordinate system generator available to the user. The coordinates may be nonorthogonal.

The metric coefficients and Jacobian describing the nonorthogonal grid transformation are computed internally by *Proteus*. This calculation involves numerically computing first derivatives of the user-specified coordinates. Since *Proteus* solves the Navier-Stokes equations in fully conservative form, the metric coefficients themselves are factors in terms whose first and second derivatives are also computed numerically. In effect, then, third derivatives of the user-specified coordinates are used in the solution. Care should therefore be taken in ensuring that these coordinates are smooth. No coordinate smoothing is done by *Proteus* itself.

The Cartesian (x,y,z) coordinates describing the computational coordinate system are read from an unformatted file as follows:

```
 read (ngrid) ng1,ng2,ng3
 read (ngrid) (((xc(j1,j2,j3),j1=1,ng1),j2=1,ng2),j3=1,ng3),
$             (((yc(j1,j2,j3),j1=1,ng1),j2=1,ng2),j3=1,ng3),
$             (((zc(j1,j2,j3),j1=1,ng1),j2=1,ng2),j3=1,ng3)
```

The parameters read from the file are defined as follows:

NG1	Number of points in the ξ direction. The maximum value allowed is the value of the dimensioning parameter N1P. (See Section 6.2.)
NG2	Number of points in the η direction. The maximum value allowed is the value of the dimensioning parameter N2P. (See Section 6.2.)
NG3	Number of points in the ζ direction. The maximum value allowed is the value of the dimensioning parameter N3P. (See Section 6.2.)
XC	Cartesian x-coordinate.
YC	Cartesian y-coordinate.
ZC	Cartesian z-coordinate.

Note that the number of points NG1, NG2, and NG3 used to specify the computational *coordinate system* need not be the same as the number of points N1, N2, and N3 used in the computational *mesh.* The coordinates of the points in the computational mesh, which is the mesh used in the *Proteus* solution, will be found by interpolation among the points in the computational coordinate system.

TABLE 3-1. - NORMALIZING CONDITIONS

Variable	Normalizing Value
Length	$L_n = L_r$
Velocity	$u_n = u_r$
Temperature	$T_n = T_r$
Density	$\rho_n = \rho_r$
Viscosity	$\mu_n = \mu_r$
Thermal conductivity	$k_n = k_r$
Pressure	$p_n = \rho_r u_r^2$
Energy per unit volume	$e_n = \rho_r u_r^2$
Gas constant	$R_n = u_r^2 / T_r$
Specific heat	$c_{p_n} = u_r^2 / T_r$
Enthalpy	$h_n = u_r^2$
Time	$t_n = L_r / u_r$
Turbulent kinetic energy	$k_n = u_r^2$
Turbulent dissipation rate	$\varepsilon_n = \rho_r u_r^4 / \mu_r$

TABLE 3-2. - REFERENCE CONDITIONS

Variable	Reference Value
Length	L_r
Velocity	u_r
Temperature	T_r
Density	ρ_r
Viscosity	μ_r
Thermal conductivity	k_r
Pressure	$p_r = \rho_r \bar{R} T_r / g_c$
Energy per unit volume	$e_r = \rho_r u_r^2$
Enthalpy	u_r^2
Specific heat	u_r^2 / T_r
Time	L_r / u_r
Turbulent kinetic energy	u_r^2
Turbulent dissipation rate	$\rho_r u_r^4 / \mu_r$

TABLE 3-3. - OUTPUT VARIABLES

IVOUT	VARIABLE	DEFINITION
		Velocities
1	x-velocity	u
2	y-velocity	v
3	z-velocity	w
4	Mach number	$M = \frac{\lvert\vec{V}\rvert}{a}$
5	Speed of sound	$a = \sqrt{\gamma R T}$
6	Contravariant velocity normal to ξ surface	$(\xi_t + u\xi_x + v\xi_y + w\xi_z)/(\xi_x^2 + \xi_y^2 + \xi_z^2)^{1/2}$
7	Contravariant velocity normal to η surface	$(\eta_t + u\eta_x + v\eta_y + w\eta_z)/(\eta_x^2 + \eta_y^2 + \eta_z^2)^{1/2}$
8	Contravariant velocity normal to ζ surface	$(\zeta_t + u\zeta_x + v\zeta_y + w\zeta_z)/(\zeta_x^2 + \zeta_y^2 + \zeta_z^2)^{1/2}$
9	Total velocity magnitude	$\lvert\vec{V}\rvert = (u^2 + v^2 + w^2)^{1/2}$
10	x-momentum	ρu
11	y-momentum	ρv
12	z-momentum	ρw
13	ξ-velocity	$V_\xi = (x_\xi u + y_\xi v + z_\xi w)/(x_\xi^2 + y_\xi^2 + z_\xi^2)^{1/2}$
14	η-velocity	$V_\eta = (x_\eta u + y_\eta v + z_\eta w)/(x_\eta^2 + y_\eta^2 + z_\eta^2)^{1/2}$
15	ζ-velocity	$V_\zeta = (x_\zeta u + y_\zeta v + z_\zeta w)/(x_\zeta^2 + y_\zeta^2 + z_\zeta^2)^{1/2}$
16	Flow angle α_v, deg.	$\tan^{-1}\frac{v}{u}$
17	Flow angle α_w, deg.	$\tan^{-1}\frac{w}{u}$
		Densities
20	Static density	ρ
21	Total density	$\rho_T = \rho\left(1 + \frac{\gamma - 1}{2}M^2\right)^{1/(\gamma-1)}$

IVOUT	VARIABLE	DEFINITION
	Pressures	
30	Static pressure	p
31	Total pressure	$p_T = p\left(1 + \frac{\gamma - 1}{2} M^2\right)^{\gamma/(\gamma-1)}$
32	Static pressure coefficient	$c_p = \frac{\bar{p} - p_r}{\rho_r u_r^2/2g_c}$
33	Total pressure coefficient	$c_{p_T} = \frac{\bar{p}_T - p_{T_r}}{\rho_r u_r^2/2g_c}$
34	Pitot pressure	$p_p = p_T$ if $M \le 1$ $p_p = p\left(\frac{\gamma + 1}{2} M^2\right)^{\gamma/(\gamma-1)} \cdot \left(\frac{2\gamma}{\gamma + 1} M^2 - \frac{\gamma - 1}{\gamma + 1}\right)^{-1/(\gamma-1)}$ if $M > 1$
35	Dynamic pressure	$\frac{1}{2}\rho(u^2 + v^2 + w^2)\frac{\rho_r u_r^2}{g_c p_r}$
	Temperatures	
40	Static temperature	T
41	Total temperature	$T_T = T\left(1 + \frac{\gamma - 1}{2} M^2\right)$
	Energies	
50	Total energy per unit volume	E_T
51	Total energy	$\frac{E_T}{\rho}$
52	Internal energy	$e_i = c_v T$
53	Kinetic energy	$e_k = \frac{1}{2}(u^2 + v^2 + w^2)$
	Enthalpies	
60	Static enthalpy	$h = c_p T$
61	Total enthalpy	$h_T = c_p T_T$

IVOUT	VARIABLE	DEFINITION
	Vorticities	
70	x-vorticity	$\Omega_x = \frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}$
71	y-vorticity	$\Omega_y = \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}$
72	z-vorticity	$\Omega_z = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$
73	Total vorticity magnitude	$\lvert\vec{\Omega}\rvert = (\Omega_x^2 + \Omega_y^2 + \Omega_z^2)^{1/2}$
	Entropies	
80	Entropy	$s = \bar{c}_v \ln\left(\frac{\bar{p}}{p_r}\right) + \bar{c}_p \ln\left(\frac{\rho_r}{\bar{\rho}}\right)$
	Temperature-Dependent Parameters	
90	Laminar viscosity coefficient	$\mu_l = \mu - \mu_t$
91	Laminar second coefficient of viscosity	$\lambda_l = -\frac{2\mu_l}{3}$
92	Laminar thermal conductivity coefficient	$k_l = k - k_t$
93	Specific heat at constant pressure	$\bar{c}_p$
94	Specific heat at constant volume	$\bar{c}_v$
95	Ratio of specific heats	$\gamma = \frac{c_p}{c_v}$

IVOUT	VARIABLE	DEFINITION
	Turbulence Parameters	
100	Turbulent viscosity coefficient	μ_t
101	Turbulent second coefficient of viscosity	$\lambda_t = -\frac{2\mu_t}{3}$
102	Turbulent thermal conductivity coefficient	$k_t = \frac{\bar{c}_p \bar{\mu}_t}{Pr_t} \frac{1}{k_r}$
103	Effective viscosity coefficient	μ
104	Effective second coefficient of viscosity	λ
105	Effective thermal conductivity coefficient	k
106	Turbulent kinetic energy	k
107	Turbulent dissipation rate	ε
108	Inner region coordinate	$y^+ = \frac{\rho_w u_\tau y_n}{\mu_w} Re_r$
109	Inner region velocity	$u^+ = \frac{\lvert\vec{V}\rvert}{u_\tau} = \lvert\vec{V}\rvert \left(\frac{1}{Re_r} \frac{\mu}{\rho} \lvert\vec{\Omega}\rvert \right)_w^{-1/2}$
	Gradients	
120	Shear stress	τ_{xx}
121	Shear stress	τ_{yy}
122	Shear stress	τ_{zz}
123	Shear stress	τ_{xy}
124	Shear stress	τ_{xz}
125	Shear stress	τ_{yz}
126	Heat flux	q_x
127	Heat flux	q_y
128	Heat flux	q_z

IVOUT	VARIABLE	DEFINITION
	Coordinates	
200	Cartesian x-coordinate	x
201	Cartesian y-coordinate	y
202	Cartesian z-coordinate	z
203	Local Δx	$\lvert J^{-1}\rvert\left[(\eta_y\zeta_z-\eta_z\zeta_y)\Delta\xi+(\xi_z\zeta_y-\xi_y\zeta_z)\Delta\eta+(\xi_y\eta_z-\xi_z\eta_y)\Delta\zeta\right]$
204	Local Δy	$\lvert J^{-1}\rvert\left[(\eta_z\zeta_x-\eta_x\zeta_z)\Delta\xi+(\xi_x\zeta_z-\xi_z\zeta_x)\Delta\eta+(\xi_z\eta_x-\xi_x\eta_z)\Delta\zeta\right]$
205	Local Δz	$\lvert J^{-1}\rvert\left[(\eta_x\zeta_y-\eta_y\zeta_x)\Delta\xi+(\xi_y\zeta_x-\xi_x\zeta_y)\Delta\eta+(\xi_x\eta_y-\xi_y\eta_x)\Delta\zeta\right]$
206	Local cell Reynolds number Re_c	$Re_r\dfrac{\rho\lvert\vec{V}\rvert\left[(\Delta x)^2+(\Delta y)^2+(\Delta z)^2\right]^{1/2}}{\mu}$
207	Local x-direction cell Reynolds number $(Re_c)_x$	$Re_r\dfrac{\rho\lvert u\rvert\Delta x}{\mu}$
208	Local y-direction cell Reynolds number $(Re_c)_y$	$Re_r\dfrac{\rho\lvert v\rvert\Delta y}{\mu}$
209	Local z-direction cell Reynolds number $(Re_c)_z$	$Re_r\dfrac{\rho\lvert w\rvert\Delta z}{\mu}$
	Metric Parameters	
210	Inverse of the grid transformation Jacobian	J^{-1}
211	Metric coefficient	ξ_t
212	Metric coefficient	ξ_x
213	Metric coefficient	ξ_y
214	Metric coefficient	ξ_z
215	Metric coefficient	η_t
216	Metric coefficient	η_x
217	Metric coefficient	η_y
218	Metric coefficient	η_z
219	Metric coefficient	ζ_t
220	Metric coefficient	ζ_x
221	Metric coefficient	ζ_y
222	Metric coefficient	ζ_z

IVOUT	VARIABLE	DEFINITION
		Times
230	Time step size	$\Delta\tau$
231	Time	τ
232	Local CFL number	$\Delta\tau\left\{\frac{\lvert\xi_t + u\xi_x + v\xi_y + w\xi_z\rvert}{\Delta\xi} + \frac{\lvert\eta_t + u\eta_x + v\eta_y + w\eta_z\rvert}{\Delta\eta} + \frac{\lvert\zeta_t + u\zeta_x + v\zeta_y + w\zeta_z\rvert}{\Delta\zeta} + a\left[\left(\frac{\xi_x}{\Delta\xi} + \frac{\eta_x}{\Delta\eta} + \frac{\zeta_x}{\Delta\zeta}\right)^2 + \left(\frac{\xi_y}{\Delta\xi} + \frac{\eta_y}{\Delta\eta} + \frac{\zeta_y}{\Delta\zeta}\right)^2 + \left(\frac{\xi_z}{\Delta\xi} + \frac{\eta_z}{\Delta\eta} + \frac{\zeta_z}{\Delta\zeta}\right)^2\right]^{1/2}\right\}$
233	Local ξ-direction CFL number	$\Delta\tau\left[\frac{\lvert\xi_t + u\xi_x + v\xi_y + w\xi_z\rvert}{\Delta\xi} + a\frac{(\xi_x^2 + \xi_y^2 + \xi_z^2)^{1/2}}{\Delta\xi}\right]$
234	Local η-direction CFL number	$\Delta\tau\left[\frac{\lvert\eta_t + u\eta_x + v\eta_y + w\eta_z\rvert}{\Delta\eta} + a\frac{(\eta_x^2 + \eta_y^2 + \eta_z^2)^{1/2}}{\Delta\eta}\right]$
235	Local ζ-direction CFL number	$\Delta\tau\left[\frac{\lvert\zeta_t + u\zeta_x + v\zeta_y + w\zeta_z\rvert}{\Delta\zeta} + a\frac{(\zeta_x^2 + \zeta_y^2 + \zeta_z^2)^{1/2}}{\Delta\zeta}\right]$

TABLE 3-4. - EQUATIONS SOLVED

IHSTAG	NEQ	Order of Equations	Order of Dependent Variables
1	4	Continuity, x-momentum, y-momentum, z-momentum	ρ, ρu, ρv, ρw
0	5	Continuity, x-momentum, y-momentum, z-momentum, energy	ρ, ρu, ρv, ρw, E_T

TABLE 3-5. - BOUNDARY TYPES

KBC VALUE[a]	BOUNDARY TYPE	JBC VALUES SET	EQUATIONS
± 1	No-slip adiabatic wall	11, 21, 31, ± 43, ± 53	$u = v = w = 0$, $\partial p/\partial n = \partial T/\partial n = 0$
± 2	No-slip wall, specified temperature	11, 21, 31, ± 43, 50	$u = v = w = 0$, $\partial p/\partial n = 0$, $\Delta T = 0$
± 3	Inviscid wall	± 43, ± 53, 71, m, n[b]	$\partial p/\partial n = \partial T/\partial n = 0$, $V_n = 0$, $\partial^2 V_{t1}/\partial \phi^2 = 0$, $\partial^2 V_{t2}/\partial \phi^2 = 0$
10	Subsonic inflow, linear extrapolation	14, 24, 34, 46, 56	$\partial^2 u/\partial \phi^2 = \partial^2 v/\partial \phi^2 = \partial^2 w/\partial \phi^2 = 0$, $\Delta p_T = \Delta T_T = 0$
± 11	Subsonic inflow, zero gradient	± 12, ± 22, ± 32, 46, 56	$\partial u/\partial \phi = \partial v/\partial \phi = \partial w/\partial \phi = 0$, $\Delta p_T = \Delta T_T = 0$
20	Subsonic outflow, linear extrapolation	14, 24, 34, 40, 54	$\partial^2 u/\partial \phi^2 = \partial^2 v/\partial \phi^2 = \partial^2 w/\partial \phi^2 = 0$, $\Delta p = 0$, $\partial^2 T/\partial \phi^2 = 0$
± 21	Subsonic outflow, zero gradient	± 12, ± 22, ± 32, 40, ± 52	$\partial u/\partial \phi = \partial v/\partial \phi = \partial w/\partial \phi = 0$, $\Delta p = 0$, $\partial T/\partial \phi = 0$
30	Supersonic inflow	10, 20, 30, 40, 50	$\Delta u = \Delta v = \Delta w = \Delta p = \Delta T = 0$
40	Supersonic outflow, linear extrapolation	14, 24, 34, 44, 54	$\partial^2 u/\partial \phi^2 = \partial^2 v/\partial \phi^2 = \partial^2 w/\partial \phi^2 = 0$, $\partial^2 p/\partial \phi^2 = \partial^2 T/\partial \phi^2 = 0$
± 41	Supersonic outflow, zero gradient	± 12, ± 22, ± 32, ± 42, ± 52	$\partial u/\partial \phi = \partial v/\partial \phi = \partial w/\partial \phi = 0$, $\partial p/\partial \phi = \partial T/\partial \phi = 0$
± 50	Symmetry	± 43, ± 53, 71, m, n[c]	$\partial p/\partial n = \partial T/\partial n = 0$, $V_n = 0$, $\partial V_{t1}/\partial \phi = 0$, $\partial V_{t2}/\partial \phi = 0$
60	Spatially periodic		$\mathbf{Q}_1 = \mathbf{Q}_{N_1}$, $\mathbf{Q}_1 = \mathbf{Q}_{N_2}$, or $\mathbf{Q}_1 = \mathbf{Q}_{N_3}$

[a] Use the "+" sign for 2-point one-sided differencing of first derivatives, and the "−" sign for 3-point differencing of first derivatives.

[b] The JBC values used are 84 and 89 for a ξ boundary, 79 and 89 for an η boundary, and 79 and 84 for a ζ boundary.

[c] The JBC values used are ± 83 and ± 88 for a ξ boundary, ± 78 and ± 88 for an η boundary, and ± 78 and ± 83 for a ζ boundary.

TABLE 3-6. - BOUNDARY CONDITION TYPES

JBC OR IBC VALUE[a]	EQUATION	DESCRIPTION
	Conservation Variable Boundary Conditions	
0	$\Delta Q = 0$	No change from initial conditions.
1	$Q = f$	Specified conservation variable.
± 2	$\partial Q/\partial \phi = f$	Specified coordinate direction gradient.
± 3	$\partial Q/\partial n = f$	Specified normal direction gradient[b].
4	$\partial^2 Q/\partial \phi^2 = 0$	Linear extrapolation.
5		Not used.
6		Not used.
7		Not used.
8		Not used.
9		Not used.
	x-Velocity Boundary Conditions	
10	$\Delta u = 0$	No change from initial conditions.
11	$u = f$	Specified x-velocity.
± 12	$\partial u/\partial \phi = f$	Specified coordinate direction gradient.
± 13	$\partial u/\partial n = f$	Specified normal direction gradient[b].
14	$\partial^2 u/\partial \phi^2 = 0$	Linear extrapolation.
15		Not used.
16		Not used.
17		Not used.
18		Not used.
19		Not used.
	y-Velocity Boundary Conditions	
20	$\Delta v = 0$	No change from initial conditions.
21	$v = f$	Specified y-velocity.
± 22	$\partial v/\partial \phi = f$	Specified coordinate direction gradient.
± 23	$\partial v/\partial n = f$	Specified normal direction gradient[b].
24	$\partial^2 v/\partial \phi^2 = 0$	Linear extrapolation.
25		Not used.
26		Not used.
27		Not used.
28		Not used.
29	$\tan^{-1}(v/u) = f$	Specified flow angle in degrees.
	z-Velocity Boundary Conditions	
30	$\Delta w = 0$	No change from initial conditions.
31	$w = f$	Specified z-velocity.
± 32	$\partial w/\partial \phi = f$	Specified coordinate direction gradient.
± 33	$\partial w/\partial n = f$	Specified normal direction gradient[b].
34	$\partial^2 w/\partial \phi^2 = 0$	Linear extrapolation.
35		Not used.
36		Not used.
37		Not used.
38		Not used.
39	$\tan^{-1}(w/u) = f$	Specified flow angle in degrees.

JBC OR IBC VALUE[a]	EQUATION	DESCRIPTION
		Pressure Boundary Conditions
40	$\Delta p = 0$	No change from initial conditions.
41	$p = f$	Specified static pressure.
± 42	$\partial p/\partial \phi = f$	Specified coordinate direction gradient.
± 43	$\partial p/\partial n = f$	Specified normal direction gradient[b].
44	$\partial^2 p/\partial \phi^2 = 0$	Linear extrapolation.
45		Not used.
46	$\Delta p_T = 0$	No change from initial conditions.
47	$p_T = f$	Specified total pressure.
48		Not used.
49		Not used.
		Temperature Boundary Conditions
50	$\Delta T = 0$	No change from initial conditions.
51	$T = f$	Specified static temperature.
± 52	$\partial T/\partial \phi = f$	Specified coordinate direction gradient.
± 53	$\partial T/\partial n = f$	Specified normal direction gradient[b].
54	$\partial^2 T/\partial \phi^2 = 0$	Linear extrapolation.
55		Not used.
56	$\Delta T_T = 0$	No change from initial conditions.
57	$T_T = f$	Specified total temperature.
58		Not used.
59		Not used.
		Density Boundary Conditions
60	$\Delta \rho = 0$	No change from initial conditions.
61	$\rho = f$	Specified static density.
± 62	$\partial \rho/\partial \phi = f$	Specified coordinate direction gradient.
± 63	$\partial \rho/\partial n = f$	Specified normal direction gradient[b].
64	$\partial^2 \rho/\partial \phi^2 = 0$	Linear extrapolation.
65		Not used.
66		Not used.
67		Not used.
68		Not used.
69		Not used.

JBC OR IBC VALUE[a]	EQUATION	DESCRIPTION
	Normal[c] and Coordinate Direction Velocity Boundary Conditions	
70		Not used.
71	$V_n = f$	Specified normal velocity.
±72	$\partial V_n / \partial \phi = f$	Specified coordinate direction gradient.
±73	$\partial V_n / \partial n = f$	Specified normal direction gradient[b].
74	$\partial^2 V_n / \partial \phi^2 = 0$	Linear extrapolation.
75		Not used.
76	$V_\xi = f$	Specified ξ-velocity.
±77	$\partial V_\xi / \partial \phi = f$	Specified coordinate direction gradient.
±78	$\partial V_\xi / \partial n = f$	Specified normal direction gradient[b].
79	$\partial^2 V_\xi / \partial \phi^2 = 0$	Linear extrapolation.
80		Not used.
81	$V_\eta = f$	Specified η-velocity.
±82	$\partial V_\eta / \partial \phi = f$	Specified coordinate direction gradient.
±83	$\partial V_\eta / \partial n = f$	Specified normal direction gradient[b].
84	$\partial^2 V_\eta / \partial \phi^2 = 0$	Linear extrapolation.
85		Not used.
86	$V_\zeta = f$	Specified ζ-velocity.
±87	$\partial V_\zeta / \partial \phi = f$	Specified coordinate direction gradient.
±88	$\partial V_\zeta / \partial n = f$	Specified normal direction gradient[b].
89	$\partial^2 V_\zeta / \partial \phi^2 = 0$	Linear extrapolation.
	User-Supplied Boundary Conditions	
90	$\Delta F = 0$	No change from initial conditions.
91	$F = f$	Specified function.
±92	$\partial F / \partial \phi = f$	Specified coordinate direction gradient.
±93	$\partial F / \partial n = f$	Specified normal direction gradient[b].
94	$\partial^2 F / \partial \phi^2 = 0$	Linear extrapolation.
95		Not used.
96		Not used.
97		Not used.
98		Not used.
99		Not used.

[a] Use the "+" sign for 2-point one-sided differencing, and the "−" sign for 3-point one-sided differencing.

[b] Normal derivatives are positive in the direction of increasing ξ or η.

[c] Normal velocity is positive in the direction of increasing ξ or η.

TABLE 3-7. - BOUNDARY CONDITION TYPES FOR k-ε EQUATIONS

JBCT OR IBCT VALUE[a]	EQUATION	DESCRIPTION
	Turbulent Kinetic Energy Boundary Conditions	
0	$\Delta k = 0$	No change from initial conditions.
1	$k = f$	Specified turbulent kinetic energy.
± 2	$\partial k / \partial \phi = f$	Specified coordinate direction gradient.
3		Not used.
4	$\partial^2 k / \partial \phi^2 = 0$	Linear extrapolation.
5		Not used.
6		Not used.
7		Not used.
8		Not used.
9		Not used.
	Turbulent Dissipation Rate Boundary Conditions	
10	$\Delta \varepsilon = 0$	No change from initial conditions.
11	$\varepsilon = f$	Specified turbulent dissipation rate.
± 12	$\partial \varepsilon / \partial \phi = f$	Specified coordinate direction gradient.
13		Not used.
14	$\partial^2 \varepsilon / \partial \phi^2 = 0$	Linear extrapolation.
15		Not used.
16		Not used.
17		Not used.
18		Not used.
19		Not used.

[a] Use the "+" sign for 2-point one-sided differencing, and the "−" sign for 3-point one-sided differencing.

4.0 OUTPUT DESCRIPTION

Several output files may be created during a *Proteus* run. The standard output is a formatted file written to Fortran unit NOUT that is intended for printing. Additional unformatted files may be written for use as input by various post-processing programs. Unformatted restart files may also be written for use as input for a subsequent *Proteus* run.

4.1 STANDARD OUTPUT

The standard *Proteus* output is a formatted file written to Fortran unit NOUT, and is intended for printing. Actual examples of typical standard output files are presented in Section 9.0. Unless specified otherwise, all of the output parameters in the standard output are nondimensional, with the appropriate reference condition from Table 3-2 as the nondimensionalizing factor.

4.1.1 Title Page and Namelists

The standard *Proteus* output begins with a title page,[12] which identifies the version of *Proteus* being run and lists the user-specified title for the run. This is followed by a printout of the contents of the input namelists RSTRT, IO, GMTRY, FLOW, BC, NUM, TIME, and TURB. Note that, for variables not specified by the user in the input namelists, the values in this printout will be the default values.

4.1.2 Normalizing and Reference Conditions

The dimensional values for the normalizing and reference conditions are printed on the next page, with the appropriate units as set by the input parameter IUNITS. The normalizing conditions are the parameters used to nondimensionalize the governing equations. The reference conditions are used during input and output for nondimensionalization of various parameters and for specifying various flow conditions. The distinction between normalizing and reference conditions is described in greater detail in Section 3.1.1. They are listed in Tables 3-1 and 3-2.

After the printout of the normalizing and reference parameters comes anything written to unit NOUT by the user-supplied subroutine INIT. For the default version of INIT supplied with *Proteus*, this consists only of the contents of namelist IC.

4.1.3 Boundary Conditions

The next page is a printout of the boundary conditions being used. The boundary condition parameters JBC and GBC are printed for the ξ, η, and ζ boundaries. If the Chien two-equation turbulence model is being used, the boundary condition parameters for the k-ε equations, JBCT and GBCT, are printed next. If time-dependent boundary conditions are being used, this is followed by a listing of the input tables of GTBC vs. NTBCA.

4.1.4 Computed Flow Field

The bulk of the standard *Proteus* output consists of printout of the computed flow field. The input array IVOUT determines which variables are printed, as described in Section 3.1.4. The variables currently available for printing are listed and defined in Table 3-3. The printout for each variable at a given time level will begin on a separate page. The header for each variable will include the time level n, and, for global time steps (IDTAU = 1 - 4, 7), the time t and time increment Δt in seconds. The flow field results are printed

[12] In this discussion, when "pages" of output are referred to it is assumed that the file is printed with Fortran carriage control in effect.

in blocks, with each block corresponding to a ζ location. Within each block, each column corresponds to a ξ location, and each row to an η location. The columns and rows are numbered with the ξ and η indices.

Flow field results are printed at time levels and grid points specified by the user through parameters in namelist IO. Since this printout can be very lengthy, the user is encouraged to minimize the amount of printed output by making judicious use of these parameters. Usually, the computed results can be examined most efficiently using post-processing graphics routines like CONTOUR or PLOT3D. (See Section 4.2).

After the flow field printout, if the run ends normally a message is printed indicating whether or not the calculation converged.

4.1.5 Boundary Parameters

After each flow field printout, various parameters may be printed along the boundaries. The input arrays IWOUT1, IWOUT2, and IWOUT3 determine whether or not these parameters are printed, as described in Section 3.1.4. The parameters printed are defined below. Note that some of these are meaningful only if the boundary is a solid wall.

X, Y, Z	Cartesian coordinates x, y, and z.
P	Static pressure p.
CF	Skin friction coefficient c_f, defined as $$c_f = \frac{\bar{\mu}\dfrac{\partial \bar{V}_t}{\partial \bar{n}}}{\dfrac{1}{2}\rho_r u_r^2} = \frac{2}{Re_r}\mu\frac{\partial V_t}{\partial n}$$ where the overbar denotes a dimensional quantity. In this equation $\partial V_t/\partial n$ represents the normal derivative of the tangential velocity.
TAUW	Shear stress τ_w, defined as $$\tau_w = \mu\frac{\partial V_t}{\partial n}$$ τ_w is thus nondimensionalized by $\mu_r u_r/L_r$.
T	Static temperature T.
QW	Heat flux q_w, defined as $$q_w = -k\frac{\partial T}{\partial n}$$ In this equation $\partial T/\partial n$ represents the normal derivative of the temperature. q_w is thus nondimensionalized by $k_r T_r/L_r$.
H	Heat transfer coefficient h, defined as $$h = \frac{q_w}{T-1} = \frac{-k\dfrac{\partial T}{\partial n}}{T-1}$$ h is thus nondimensionalized by k_r/L_r.
ST	Stanton number St, defined as $$St = \frac{\bar{h}}{\rho_r u_r \bar{c}_p} = \frac{h}{c_p}\frac{1}{Re_r Pr_r}$$

The boundary parameters are printed at the same time levels as the flow field printout, and at the points on the boundary specified by the input parameters IPRT1, IPRT2, and IPRT3, or IPRT1A, IPRT2A, and IPRT3A.

For this printout, normal derivatives are computed using a normal vector $\vec{n}$ directed into the flow field. This means, for example, that the skin friction c_f will be positive for attached flow, even on the upper boundary.

4.1.6 Convergence History

In evaluating the results of a steady *Proteus* calculation, it's important to consider the level of convergence. This may be done by examining one of the forms of the residual for each equation. The residual is simply the number resulting from evaluating the steady form of the equation at a specific grid point and time (or iteration) level. Ideally, the residuals would all approach zero at convergence. In practice, however, for "real" problems they often drop to a certain level and then level off. Continuing the calculation beyond this point will not improve the results.

A decrease in the L_2 norm of the residual of three orders of magnitude is sometimes considered sufficient. Convergence, however, is in the eye of the beholder. The amount of decrease in the residual necessary for convergence will vary from problem to problem. For some problems, it may even be more appropriate to measure convergence by some flow-related parameter, such as the lift coefficient for an airfoil. Determining when a solution is sufficiently converged is, in some respects, a skill best acquired through experience.

At the end of a *Proteus* calculation, if first-order time differencing and steady boundary conditions were used, a summary of the convergence history is printed for each governing equation.[13] The parameters in this printout are defined as follows:

LEVEL	Time level n.
CHGMAX	Maximum change in absolute value of the dependent variables from time level $n-1$ to n.[14] $$\Delta \mathbf{Q}_{max} = \max \left| \Delta \mathbf{Q}^{n-1}_{i,j,k} \right|$$
CHGAVG	Maximum change in absolute value of the dependent variables, averaged over the last NITAVG time steps.[14] $$\Delta \mathbf{Q}_{avg} = \frac{1}{\text{NITAVG}} \sum_{m=n-\text{NITAVG}}^{n} \Delta \mathbf{Q}^{m-1}_{max}$$
RESL2	The L_2 norm of the residual at time level n. $$\mathrm{R}_{L_2} = \left(\sum (\mathrm{R}^n_{i,j,k})^2 \right)^{1/2}$$
RESAVG	The average absolute value of the residual at time level n, R_{avg}.
RESMAX	The maximum absolute value of the residual at time level n, R_{max}.
LRMAX	The grid indices (i,j,k) corresponding to the location of R_{max}.

13 Second-order time differencing should be used only for unsteady problems, for which "convergence" has no meaning. It should also be noted that the computation of the residuals in the code is correct only for first-order time differencing.

14 For the energy equation, the change in $\overline{E}_T$ is divided by $E_{T_r} = \rho_r \overline{R} T_r / (\gamma_r - 1) + \rho_r u_r^2 / 2$, so that it is the same order of magnitude as the other conservation variables.

In computing the residuals, the summations, maximums, and averages are over all interior grid points, plus points on spatially periodic boundaries.

To avoid undesirably long tables, the convergence parameters are printed at an interval that limits the printout to NHMAX time levels. NHMAX can be specified by the user in namelist IO. However, the residuals are always printed at the first two time levels. This is done because the residuals at time level 1 (the initial condition level) may not be truly representative of the degree of convergence. For instance, if the initial conditions are zero velocity and constant pressure and temperature at every interior point, the computed residuals will be exactly zero. When the time marching procedure begins, however, the flow field will start developing in response to the boundary conditions, and the residuals will reach a maximum in the first few time steps. Note that, in the printout, CHGMAX will be zero until time level n = ICHECK. CHGAVG will only be computed when ICTEST = 2, and will be zero until time level n = NITAVG.

As noted in Section 8.1 of Volume 1, adding artificial viscosity changes the original governing partial differential equations. For cases run with artificial viscosity, therefore, the residuals are printed both with and without the artificial viscosity terms included. This may provide some estimate of the overall error in the solution introduced by the artificial viscosity. Convergence is determined by the residuals with the artificial viscosity terms included.

4.1.6 Additional Output

In addition to the output discussed above, various types of additional printout can be generated by the IDEBUG options, as discussed in Section 3.1.4. Various diagnostics may also appear in the standard output file. These are discussed in greater detail in Section 7.0.

4.2 PLOT FILES

The amount of flow field data generated by a Navier-Stokes code is normally much too large to efficiently comprehend by examining printed output. The computed results are therefore generally examined graphically using various post-processing plotting routines. These plotting routines require as input a file or files, generally called plot files, that are written by the flow solver and contain the coordinates and computed flow field data.

Various types of unformatted plot files may be written by *Proteus*, as controlled by the input parameter IPLOT discussed in Section 3.1.4. These files are designed for use by either the CONTOUR or PLOT3D plotting programs.[15]

CONTOUR is a three-dimensional plotting program developed at NASA Lewis using internal Lewis-developed graphics routines. It currently can be used only at NASA Lewis on the Amdahl 5860 computer using the VM operating system, or from the Scientific VAX Cluster using the VMS operating system. Originally designed for use with three-dimensional Parabolized Navier-Stokes (PNS) codes, it can be used to generate various types of contour and velocity vector plots in computational planes.

PLOT3D (Walatka, Buning, Pierce, and Elson, 1990) is a sophisticated three-dimensional plotting program specifically designed for displaying results of computational fluid dynamics analyses. It is widely used in government, industry, and universities for interactive visualization of complex flow field data generated by CFD analyses. The computational grid is stored in one file, called an XYZ file, and the computed flow field is stored in another file, called a Q file. There are several options within PLOT3D concerning the format of these files. At NASA Lewis, PLOT3D is available on the Silicon Graphics IRIS workstations and on the Scientific VAX Cluster.

It should be noted that, in Fortran, unformatted files are not generally transportable from computer to computer. If the plot files written by *Proteus* are to be used on some other computer (e.g., a graphics workstation), a separate conversion program will normally be required.

[15] If only the last computed time level is of interest, the restart files may also be used for plotting with PLOT3D. See Section 4.4.

4.2.1 CONTOUR Plot File (IPLOT = 1)

With the IPLOT = 1 option in *Proteus*, a plot file is generated for use by CONTOUR using the following Fortran statements:

```
           write (nplot) title
           write (nplot) machr,rer,lr,ur,pr,tr,rhor,rg,gamr
           do 20 il = 1,nl
           write (nplot) level,n2,n3,isym,system
           do 10 i2 = 1,n2
           write (nplot) ((f(ivar,il,i2,13),ivar=1,14),i3=1,n3)
     10    continue
     20    continue
```

All of the above WRITE statements are executed for each time level written into the file. The plot file thus consists of multiple sets of data, each containing the computed results at a single time level. Note that the value of the time τ is not written into the file. Note also that N1, the number of grid points in the ξ direction, is not written into the file.

Unless specified otherwise, all of the parameters written into the CONTOUR plot file are nondimensional, with the appropriate reference condition as the nondimensionalizing factor. The parameters are defined as follows:

TITLE	A descriptive title for the problem.
MACHR	Reference Mach number, $M_r = u_r / \sqrt{\gamma_r \bar{R} T_r}$. This is a real variable.
RER	Reference Reynolds number, $Re_r = \rho_r u_r L_r / \mu_r$.
LR	Reference length L_r in feet (meters). This is a real variable.
UR	Reference velocity u_r in ft/sec (m/sec).
PR	Reference static pressure $p_r = \rho_r \bar{R} T_r / g_c$ in lb$_f$/ft^2 (N/m^2).
TR	Reference temperature T_r in °R (K).
RHOR	Reference density ρ_r in lb$_m$/ft^3 (kg/m^3).
RG	Gas constant $\bar{R}$ in ft^2/sec^2-°R (m^2/sec^2-K).
GAMR	Reference ratio of specific heats, $\gamma_r = c_{p_r} / c_{v_r}$.
LEVEL	Time level n.
N2	Number of grid points N_2 in the η direction.
N3	Number of grid points N_3 in the ζ direction.
ISYM	A symmetry parameter used in CONTOUR, set equal to 1.
SYSTEM	A coordinate system parameter used in CONTOUR, set equal to 0.
F(1,,,)	Cartesian x coordinate.
F(2,,,)	Cartesian y coordinate.
F(3,,,)	Cartesian z coordinate.
F(4,,,)	Velocity in the ξ direction, V_ξ.
F(5,,,)	Velocity in the η direction, V_η.
F(6,,,)	Velocity in the ζ direction, V_ζ.
F(7,,,)	Static pressure $\bar{p}/p_r$.
F(8,,,)	Static temperature T.
F(9,,,)	Mach number $M = \sqrt{u^2 + v^2 + w^2} / \sqrt{\gamma R T}$

F(10,,,)	x-velocity u.
F(11,,,)	y-velocity v.
F(12,,,)	z-velocity w.
F(13,,,)	Static density ρ.
F(14,,,)	Total energy per unit volume E_T.

4.2.2 PLOT3D/WHOLE Plot Files (IPLOT = 2)

With the IPLOT = 2 option in *Proteus*, the XYZ and Q files are written in PLOT3D/WHOLE format. With this option, the XYZ file is written using the following Fortran statements:

```
      write (nplotx) n1,n2,n3
      write (nplotx) (((x(i1,i2,i3),i1=1,n1),i2=1,n2),i3=1,n3),
     $               (((y(i1,i2,i3),i1=1,n1),i2=1,n2),i3=1,n3),
     $               (((z(i1,i2,i3),i1=1,n1),i2=1,n2),i3=1,n3)
```

The Q file is written using the following Fortran statements:

```
      write (nplot) n1,n2,n3
      write (nplot) machr,aoa,rer,tau(1,1,1)
      write (nplot) ((((qplot(i1,i2,i3,ivar),i1=1,n1),i2=1,n2),
     $                                   i3=1,n3),ivar=1,5)
```

The above WRITE statements for the Q file are executed for each time level written into the file. The Q file thus consists of multiple sets of data, each containing the computed results at a single time level. The XYZ file is written only once.[16]

The parameters written into the file are defined as follows:

N1	Number of grid points N_1 in the ξ direction.
N2	Number of grid points N_2 in the η direction.
N3	Number of grid points N_3 in the ζ direction.
X	Cartesian x coordinate.
Y	Cartesian y coordinate.
Z	Cartesian z coordinate.
MACHR	Reference Mach number, $M_r = u_r/\sqrt{\gamma_r \bar{R} T_r}$. This is a real variable.
AOA	In PLOT3D, the angle of attack. Set equal to 0. in *Proteus*.
RER	Reference Reynolds number, $Re_r = \rho_r u_r L_r/\mu_r$.
TAU(1,1,1)	The time $\tau_{1,1,1}$.[17]
QPLOT(,,,1)	Static density ρ.
QPLOT(,,,2)	x-momentum $\rho u M_r$.
QPLOT(,,,3)	y-momentum $\rho v M_r$.
QPLOT(,,,4)	z-momentum $\rho w M_r$.
QPLOT(,,,5)	Total energy per unit volume $E_T M_r^2$.

[16] The current version of PLOT3D does not work for multiple time levels, although future versions might.

[17] Note that with IDTAU = 5 or 6, τ will vary in space, and therefore $\tau_{i,j,k} \neq \tau_{1,1,1}$.

All of the parameters written into the XYZ and Q files are nondimensional. However, PLOT3D assumes that velocity is nondimensionalized by the reference speed of sound $a_r = (\gamma_r \bar{R} T_r)^{1/2}$, and that energy is nondimensionalized by $\rho_r a_r^2$. In *Proteus* these variables are nondimensionalized by u_r and $\rho_r u_r^2$. That is why M_r appears in the definitions of QPLOT(,,,2) through QPLOT(,,,5).

4.2.3 PLOT3D/PLANES Plot Files (IPLOT = 3)

With the IPLOT = 3 option in *Proteus*, the XYZ and Q files are written in PLOT3D/PLANES format. With this option, the XYZ file is written using the following Fortran statements:

```
            write (nplotx) n1,n2,n3
            do 10 i3 = 1,n3
            write (nplotx) ((x(i1,i2,i3),i1=1,n1),i2=1,n2),
           $               ((y(i1,i2,i3),i1=1,n1),i2=1,n2),
           $               ((z(i1,i2,i3),i1=1,n1),i2=1,n2)
   10       continue
```

The Q file is written using:

```
            write (nplot) n1,n2,n3
            write (nplot) machr,aoa,rer,tau(1,1,1)
            do 20 i3 = 1,n3
            write (nplot) (((qplot(i1,i2,i3,ivar),i1=1,n1),i2=1,n2),
           $                                         ivar=1,5)
   20       continue
```

As in the IPLOT = 2 option, the above WRITE statements for the Q file are executed for each time level written into the file. The Q file thus consists of multiple sets of data, each containing the computed results at a single time level. The XYZ file is written only once.

4.3 CONVERGENCE HISTORY FILE

In Section 4.1.6, the convergence history printout is described. The information in this printout is read from an unformatted convergence history file that is updated whenever convergence is checked during a *Proteus* calculation. Plotting routines are also available at NASA Lewis to plot any of the convergence parameters as a function of time level.

The file is written in subroutine RESID using the following Fortran statements. At the first time step of the run,

```
            write (nhist) n1,n2,n3,neq,idtau,ictest,nitavg,ihstag,
           $              iav2e,iav4e,ur,lr,(eps(ieq),ieq=1,neq)
```

Then, at each time level that convergence is being checked,

```
            write (nhist) it,tau(1,1,1),icheck
            write (nhist) (chgmax(ieq,1),chgavg(ieq),resl2(ieq,1),
           $               resavg(ieq,1),resmax(ieq,1),lrmax(1,ieq,1),
           $               lrmax(2,ieq,1),lrmax(3,ieq,1),ieq=1,neq)
```

Finally, again at each time level that convergence is being checked, but only for cases run with explicit artificial viscosity,

```
            write (nhist) (chgmax(ieq,1),chgavg(ieq),resl2(ieq,2),
           $               resavg(ieq,1),resmax(ieq,1),lrmax(1,ieq,2),
           $               lrmax(2,ieq,2),lrmax(3,ieq,2),ieq=1,neq)
```

The parameters written into the file are defined as follows:

N1	Number of grid points N_1 in the ξ direction.
N2	Number of grid points N_2 in the η direction.

N3	Number of grid points N_3 in the ζ direction.
NEQ	The number of coupled governing equations N_{eq} being solved.
IDTAU	Flag for time step selection method.
ICTEST	Flag for type of convergence test.
NITAVG	Number of time steps used in the moving average convergence test.
IHSTAG	Flag for constant stagnation enthalpy option.
IAV2E	Flag for second-order explicit artificial viscosity.
IAV4E	Flag for fourth-order explicit artificial viscosity.
UR	Reference velocity u_r.
LR	Reference length L_r. This is a real variable.
EPS(IEQ)	Value ε used to determine convergence.
IT	Current time level n.
TAU(1,1,1)	Current value of the time marching parameter $\tau_{1,1,1}$ at $\xi = \eta = \zeta = 0$.
ICHECK	Results are checked for convergence every ICHECK'th time level.
CHGMAX(IEQ,1)	Absolute value of the maximum change in the dependent variables from time level $n-1$ to n.
CHGAVG(IEQ)	Average of the absolute value of the maximum change in the dependent variables for the last NITAVG time steps.
RESL2(IEQ,IAVR)	The L_2 norm of the residual at time level n.
RESAVG(IEQ,IAVR)	The average absolute value of the residual at time level n.
RESMAX(IEQ,IAVR)	The maximum absolute value of the residual at time level n
LRMAX(IDIR,IEQ,IAVR)	The grid indices (i, j, k) corresponding to the location of R_{max}.

In the above definitions, the subscript IEQ = 1 to N_{eq}, corresponding to the N_{eq} governing equations, IAVR = 1 or 2, corresponding to residuals computed without and with the artificial viscosity terms, and IDIR = 1, 2, or 3, corresponding to the ξ, η, and ζ directions.

4.4 RESTART FILES

It's often necessary or desirable to run a given case in a series of steps, stopping and restarting between each one. This may be done because of limitations in computer resources, or to change an input parameter. This capability is provided in *Proteus* through a restart option. With this option, the computational mesh and the computed flow field are written as unformatted output files at the end of one run, saved on a magnetic disk or tape, and read in as input files at the beginning of the next run. This process is controlled by the input parameters in namelist RSTRT. (See Section 3.1.3).

The restart files are written and read in subroutine REST. The computational mesh is written using the following Fortran statements:

```
      write (nrxout) n1,n2,n3
      write (nrxout) (((x(i1,i2,i3),i1=1,n1),i2=1,n2),i3=1,n3),
     $               (((y(i1,i2,i3),i1=1,n1),i2=1,n2),i3=1,n3),
     $               (((z(i1,i2,i3),i1=1,n1),i2=1,n2),i3=1,n3)
```

The computed flow field is written using:

```
      write (nrqout) n1,n2,n3
```

```
 write (nrqout) machr,aoa,rer,tlevel
 write (nrqout) ((((q (i1,i2,i3,ivar),i1=1,n1),i2=1,n2),
$                                    i3=1,n3),ivar=1,5)
 if (iturb .gt. 19) write (nrqout) ((((qt (i1,i2,i3,ivar),i1=1,n1),
$                                    i2=1,n2),i3=1,n3),ivar=1,2)
 write (nrqout) ((((ql(i1,i2,i3,ivar),i1=1,n1),i2=1,n2),
$                                    i3=1,n3),i3=1,n3),ivar=1,5)
 if (iturb .gt. 19) write (nrqout) ((((qtl(i1,i2,i3,ivar),i1=1,n1),
$                                    i2=1,n2),i3=1,n3),ivar=1,2)
```

The parameters written into these files are defined as follows:

N1	Number of grid points N_1 in the ξ direction.
N2	Number of grid points N_2 in the η direction.
N3	Number of grid points N_3 in the ζ direction.
X	Cartesian x coordinate.
Y	Cartesian y coordinate.
Z	Cartesian z coordinate.
MACHR	Reference Mach number, $M_r = u_r/\sqrt{\gamma_r \bar{R} T_r}$. This is a real variable.
AOA	Set equal to 0.
RER	Reference Reynolds number, $Re_r = \rho_r u_r L_r/\mu_r$.
TLEVEL	The current time level n.
Q(,,,1)	Static density ρ at time level n.
Q(,,,2)	x-momentum $\rho u M_r$ at time level n.
Q(,,,3)	y-momentum $\rho v M_r$ at time level n.
Q(,,,4)	z-momentum $\rho w M_r$ at time level n.
Q(,,,5)	Total energy per unit volume $E_T M_r^2$ at time level n.
QL(,,,1-5)	Same as Q(,,,1-5), except at time level $n-1$.
QT(,,,1)	Turbulent kinetic energy k at time level n.
QT(,,,2)	Turbulent dissipation rate ε at time level n.
QTL(,,,1-2)	Same as QT(,,,1-2), except at time level $n-1$.

Note that, except for the QT, QL, and QTL variables, these files have the same format as the XYZ and Q files created using the IPLOT = 2 option. These restart files can thus also be used as XYZ and Q files for the PLOT3D plotting program. The QT, QL, and QTL variables will not be read by PLOT3D. Note also, however, that the reverse is not true. The XYZ and Q files created using the IPLOT = 2 option may not be used as restart files, since they do not include the QT, QL, and QTL variables.[18]

[18] Actually, if the input parameters IPLT and IPLTA are such that only the final time level is written into the Q file, the XYZ and Q files may be used for an "approximate" restart. In this case, *Proteus* will set QL = Q.

PAGE ____ INTENTIONALLY BLANK

5.0 INITIAL CONDITIONS

Initial conditions are required for the mean flow equations throughout the flow field to start the time marching procedure. Although the best choice for initial conditions will be problem-dependent, some general comments can be made. First, for unsteady flows they should represent a real flow field. A converged steady-state solution from a previous run would be a good choice. Second, to minimize the number of iterations required for steady flows, the initial conditions should be reasonably close to the expected final solution. And third, to minimize problems with starting transients it is important that they represent a physically realistic flow.

If the k-ε turbulence model is being used, then initial conditions are also required for k and ε throughout the flow field. Like the mean flow equations, the best choice for the k-ε initial conditions will be highly problem-dependent, and will have great influence on the convergence and the stability of the computation. It should be noted that $k = 0$ and $\varepsilon = 0$ is merely a trivial solution to the k-ε equations (Nichols, 1990), so the calculation should not be started with $k = \varepsilon = 0$ everywhere in the flow field. The initial profiles for k and ε should be realistic and smooth, and they should at least obey the boundary conditions (i.e., approach zero at solid walls, have zero gradient in the far field, etc.) The initial conditions for the mean flow properties (p, u, v, w, and μ_t) should also be realistic for fully turbulent flows. A good place to start a calculation with the k-ε turbulence model would be a restart run from a converged solution of the same flow field with an algebraic turbulence model.

Initial conditions for the mean flow equations may be supplied by a user-written subroutine, called INIT, by a default version of INIT that specifies uniform flow with constant flow properties everywhere in the flow field, or by restart mesh and flow field files written during a previous run.

Initial conditions for the k-ε equations may be supplied by a user-written subroutine, called KEINIT, by a default version of KEINIT that computes k and ε from the mean flow field and the assumption of local equilibrium (i.e., production equals dissipation), or by restart mesh and flow field files written during a previous k-ε calculation.

5.1 USER-WRITTEN INITIAL CONDITIONS

5.1.1 Mean Flow Equations

As stated above, the best choice for initial conditions will be problem-dependent. For this reason *Proteus* does not include a general-purpose routine for setting up initial conditions. Instead, provision is made for a user-written subroutine, called INIT, that sets up the initial conditions.

The time-marching algorithm used in *Proteus* requires initial conditions for ρ, u, v, w, E_T, p, and T.[19] These variables may, of course, be computed from many different combinations of known parameters. To make this process reasonably flexible, the user may choose from several combinations exactly which variables subroutine INIT will supply. This choice is determined by the input parameter ICVARS in namelist FLOW. The following tables list the flow field variables to be supplied by subroutine INIT for the various ICVARS options, along with the Fortran variables into which they should be loaded.[20] Remember that the initial conditions must be nondimensionalized by the reference conditions listed in Table 3-2. The default value for ICVARS is 2.

19 Fewer variables may actually be needed, depending on the value of the input parameter IHSTAG.

20 Note that some input variables, like the Mach number M, are not normally saved as separate Fortran variables. To save storage they are to be loaded into existing Fortran variables in INIT. These Fortran variables will later be loaded with their normal values.

PRECEDING PAGE BLANK NOT FILMED

When the energy equation is being solved (IHSTAG = 0), the allowed values are:

ICVARS	Variables Supplied By INIT	Fortran Variables
1	$\rho, \rho u, \rho v, \rho w, E_T$	RHO, U, V, W, ET
2	p, u, v, w, T	P, U, V, W, T
3	ρ, u, v, w, T	RHO, U, V, W, T
4	p, u, v, w, ρ	P, U, V, W, RHO
5	c_p, u, v, w, T	P, U, V, W, T
6	$p, M, \alpha_v, \alpha_w, T$	P, U, V, W, T

When constant stagnation enthalpy is assumed (IHSTAG = 1), the allowed values are:

ICVARS	Variables Supplied By INIT	Fortran Variables
1	$\rho, \rho u, \rho v, \rho w$	RHO, U, V, W
2	p, u, v, w	P, U, V, W
3	ρ, u, v, w	RHO, U, V, W
5	c_p, u, v, w	P, U, V, W
6	p, M, α_v, α_w	P, U, V, W

In the above tables, c_p, α_v, and α_w represent static pressure coefficient, flow angle in degrees in the x-y plane, and flow angle in degrees in the x-z plane, respectively. These parameters are defined as:

$$c_p = \frac{\overline{p} - p_r}{\rho_r u_r^2 / 2g_c}$$

$$\alpha_v = \tan^{-1} \frac{v}{u}$$

$$\alpha_w = \tan^{-1} \frac{w}{u}$$

The *Proteus* subroutine INITC will use the variables supplied by subroutine INIT to compute ρ, u, v, w, and E_T, using perfect gas relationships if necessary. From these variables, the pressure and temperature will then be recomputed using the equation of state in subroutine EQSTAT, overwriting any values specified by the user in INIT. This ensures a consistent set of initial conditions for the time marching procedure.

Subroutine INIT is called once, immediately after the input has been read, the reference and normalizing conditions have been set, and the geometry and mesh have been defined. The user may do anything he or she desires in the subroutine, such as reading files, reading additional namelist input, making computations, etc. Any of the defined Fortran variables in common blocks may be used.[21] (Of course, they should not be changed.) The only requirement is that subroutine INIT return to the calling program (which is INITC) the combination of variables specified by ICVARS, defined at every grid point.

Subroutine INIT is also a convenient place to specify point-by-point boundary condition types and values. It's often easier to do this using Fortran coding rather than entering each value into the namelist input file.

5.1.2 k-ε Equations

As with the mean flow equations, the best choice for initial conditions for the k-ε equations will be problem-dependent. For this reason, *Proteus* does not have a general-purpose routine for setting up initial conditions for k and ε. Instead, provision is made for a user-written subroutine, called KEINIT, that supplies the initial values for k and ε.

[21] See Volume 3 for definitions of all the common block variables.

Assuming that the variables ρ, u, v, w, E_T, p, and T at time level n are available for the turbulent flow field under consideration, the k-ε turbulence model in *Proteus* also requires k, ε, and μ_t values at time levels n and $n-1$ before the k-ε calculation can be started. These variables may be computed by using many different assumptions and techniques, or simply by fitting the available experimental data.

Subroutine KEINIT is used when a calculation is first started from an initial flow field, or when a restart is made from an algebraic turbulence model calculation. It is called once, immediately after the initial conditions for the mean flow have been computed. Subroutine KEINIT must supply the initial values for the Fortran variables KE, E, MUT (omit MUT if the calculation is a restart from a previous algebraic turbulence model calculation), KEL, EL, and MUTL at every grid point. Usually, KEL, EL, and MUTL can be set equal to KE, E, and MUT, respectively. The default version of KEINIT, described below, is a good place to start when developing a user-written version.

5.2 DEFAULT INITIAL CONDITIONS

5.2.1 Mean Flow Equations

A default version of subroutine INIT is built into *Proteus* that specifies uniform flow with constant flow properties everywhere in the flow field. It uses the ICVARS = 2 option (the default value) and reads initial flow field values of p, u, v, w, and T from namelist IC. The defaults for these parameters are 1.0, 0.0, 0.0, 0.0, and 1.0, respectively, resulting in an initial flow field with $\bar{p} = p_r$, $u = v = w = 0$, and $\bar{T} = T_r$. If a value for ICVARS other than 2 is set in the input, a warning message is generated and ICVARS is reset to 2.

5.2.2 k-ε Equations

A default version of subroutine KEINIT is built into *Proteus* that calculates the k and ε initial profiles using the initial mean flow field values of ρ, u, v, w, and p, along with the assumption of local equilibrium (i.e., production equals dissipation). The turbulent viscosity value μ_t is also needed, and is obtained from the Baldwin-Lomax algebraic model. The assumption of local equilibrium has been found to give good initial values for the k profile near a solid wall. In particular, the location and magnitude of the first peak in the k profile is well predicted. Of course, k and ε will depend heavily on the initial velocity and turbulent viscosity, and unrealistic initial profiles for k and ε might result from bad initial velocity and turbulent viscosity profiles. The k and ε values calculated using this subroutine can be extremely small in regions of nearly inviscid, freestream flow. Therefore, it is up to the user to make sure that the freestream values of k and ε are appropriate for the problem under consideration, perhaps by modifying the value of the input parameter CKMIN.

5.3 RESTART INITIAL CONDITIONS

If restart mesh and flow field files were created during a previous run by setting IREST > 0 in namelist RSTRT, these files can be used to continue the calculation from the point where the previous run stopped. In this case no subroutine INIT is needed. The restart case is run by linking the existing restart mesh and flow field files to Fortran units NRXIN and NRQIN, respectively, and setting IREST = 2 or 3 in the input. New restart files will also be written to units NRXOUT and NRQOUT.

As mentioned above, subroutine KEINIT is used when the k-ε computation is done for the first time (IREST $\leq$ 1), or when restarting from a previous computation that used an algebraic turbulence model (IREST = 3). However, if the previous calculation used the k-ε turbulence model (IREST = 2), then the restart files also contain the information needed to continue the k-ε calculation. In this case, subroutine KEINIT is not needed.

When a restart case is being run, the usual namelist input described in Section 3.1 must still be read in. The following input parameters must have the same values as during the original run: IUNITS, IHSTAG, ILAMV, LR, UR, MACHR, TR, RHOR, MUR, RER, KTR, PRLR, GAMR, RG, HSTAGR, N1, N2, N3, IPACK, and SQ. The remaining input parameters either may be changed during a restart, or do not apply to a restart case. Note, however, that for many of the input parameters, such as those specifying the boundary conditions, changing values during a restart may result in temporary "re-starting" transients or even cause the calculation to blow up.

46 ____ INTENTIONALLY BLANK

6.0 RESOURCE REQUIREMENTS

Proteus was developed on the Cray X-MP and Y-MP computers at NASA Lewis Research Center. Changes that may be necessary when porting the code to another system are described in Section 6.1. Sections 6.2 and 6.3 discuss the memory and CPU time required to run the code. The values presented in these sections are for version 1.0 of the 3-D *Proteus* code, and were derived from tests run on the NASA Lewis Cray Y-MP in March, 1992. At that time the Cray was running UNICOS Version 6.0 and CFT77 5.0.2. UNICOS and CFT77 are described in the *UNICOS User Commands Reference Manual* (Cray Research, Inc., 1991), and in *CF77 Compiling System, Volume 1: Fortran Reference Manual* (Cray Research, Inc., 1990), respectively. Section 6.4 describes the size and format of the various input and output files used in the code.

6.1 COMPUTER

Proteus should be transportable to other computer systems with minimal changes. With just three known exceptions, the code is written entirely in ANSI standard Fortran 77. The first exception is the use of namelist input. With namelist input, it's relatively easy to create and/or modify input files, to read the resulting files, and to program default values. Since most Fortran compilers allow namelist input, its use is not considered a serious problem. The second exception is the use of *CALL statements to include COMDECK's, which contain the labeled common blocks, in most of the subprograms. This is a Cray UPDATE feature, and therefore the source code must be processed by UPDATE to create a file that can be compiled.[22] UPDATE is described in the *UPDATE Reference Manual* (Cray Research, Inc., 1988). Since using the *CALL statements results in cleaner, more readable code, and since many computer systems have an analogous feature, the *CALL statements were left in the program. The third exception is the use of lowercase alphabetic characters in the Fortran source code. This makes the code easier to read, and is a common extension to Fortran 77.

Several library subroutines are called by *Proteus*. SGEFA and SGESL are Cray versions of LINPACK routines. SASUM and SNRM2 are Cray Basic Linear Algebra Subprograms (BLAS). GATHER is a Cray Linear Algebra routine. ISAMAX, ISAMIN, ISRCHEQ, ISRCHFGT, ISRCHFLT, and WHENFLT are Cray search routines. TREMAIN is a Cray Fortran library routine. All of these routines are described in detail in Section 4.0 of Volume 3. In addition, SGEFA and SGESL are described in *Volume 3: UNICOS Math and Scientific Library Reference Manual* (Cray Research, Inc., 1989b) and by Dongarra, Moler, Bunch, and Stewart (1979); SASUM, SNRM2, GATHER, ISAMAX, ISAMIN, ISRCHEQ, ISRCHFGT, ISRCHFLT, and WHENFLT are described in *Volume 3: UNICOS Math and Scientific Library Reference Manual* (Cray Research, Inc., 1989b); and TREMAIN is described in *Volume 1: UNICOS Fortran Library Reference Manual* (Cray Research, Inc., 1989a). These or similar routines may be available on other systems. If not, equivalent routines will have to be coded.

6.2 MEMORY

The sizes of the dimensioned arrays in *Proteus*, and hence the amount of memory required to run the program, are set using Fortran parameter statements. These parameters are set in COMDECK PARAMS1. Larger or smaller dimensions may be set for the entire program simply by changing the appropriate parameter, and then recompiling the entire program. The parameters are defined as follows:

N1P	Maximum number of grid points in the ξ direction. The current value is 21.
N2P	Maximum number of grid points in the η direction. The current value is 21.
N3P	Maximum number of grid points in the ζ direction. The current value is 21.

[22] See the example in Section 8.1.

PRECEDING PAGE BLANK NOT FILMED

NMAXP	Maximum of N1P, N2P, and N3P.
NTOTP	Total storage required for a single three-dimensional array (i.e., N1P × N2P × N3P).
NDIAGP	Number of diagonals containing interior points in a ξ-η plane (i.e., N1P + N2P − 5).
NPNTP	Number of interior points in a ξ-η plane (i.e., (N1P − 2)(N2P − 2)).
NEQP	Number of coupled equations allowed. The number of equations to be solved depends on the value of the input parameter IHSTAG, as shown in Table 3-4 in Section 3.0. In the *Proteus* code, NEQP is initially set equal to 5. Note that *Proteus* will still work if NEQP is larger than the value of NEQ shown in Table 3-4.
NEQPM	Maximum number of coupled equations available. This is the largest permissible value for NEQP. The current value is 5.
NAMAX	Maximum number of time steps allowed in the moving average convergence test (the ICTEST = 2 option). The current value is 10.
NBC	Number of boundary conditions per equation. The current value is 2.
NTP	Maximum number of entries in the table of time-dependent boundary condition values. The current value is 10.
NTSEQP	Maximum number of time step sequences in the time step sequencing option. The current value is 10.

The total amount of memory in computer words required to run *Proteus* 3-D, compiled using CFT77, is listed in the following table for various combinations of the dimensioning parameters N1P, N2P, N3P, and NEQP.[23] On the Cray X-MP and Y-MP, each word is 64 bits long.

N1P	N2P	N3P	MEMORY	
			NEQP = 4	NEQP = 5
21	21	21	901,632	913,920
51	51	51	7,016,960	7,089,664
101	51	51	13,721,088	14,00[illegible],784

6.3 CPU TIME

Compilation of *Proteus* 3-D using the CFT77 compiler requires about 235 seconds of CPU time. The CPU time required for execution will depend on several factors, such as the turbulence model being used, and whether or not the energy equation is being solved. For the test case in Section 9.0, the CPU time ranged from about 4.8×10^{-5} to 5.7×10^{-5} seconds per grid point per time step.

6.4 INPUT/OUTPUT FILES

Several files are used in *Proteus* for various types of input and output. The contents of these files have been described previously in Sections 3.0 and 4.0. This section describes the characteristics of the files themselves. The files are identified by the Fortran variable representing the unit number. The unit numbers have default values, but all of them except NIN may be read in by the user.

Table 6-1 lists the files used in *Proteus*, giving the default unit number, briefly describing the contents of the file, and indicating when it is used. Table 6-2 summarizes the computational resources needed for each file. In this table, the record length is specified in bytes for units NIN and NOUT, and in computer words for the remaining units. The total file size is specified in printed pages for unit NOUT, and in

[23] It should be noted that version 5.0.2 of the CFT77 compiler itself requires just under 2,400,000 words.

computer words for the remaining units. Several symbols and Fortran variables are used in Table 6-2, and are defined as:

N_{av}	0 if explicit artificial viscosity is not being used, 1 if it is.
N_{turb}	0 if the input parameter ITURB = 0 or 1, 1 if ITURB = 20.
N_{eq}	The number of coupled equations being solved.
N_t	The number of time levels written into the plot file. This is determined by the input parameters IPLT and IPLTA.
N_{t1}, N_{t2}	The time level at the beginning and at the end of the calculation.
N_ξ, N_η, N_ζ	The number of grid points in the ξ, η, and ζ directions at which output is being printed. This is determined by the input parameters IPRT1, IPRT2, IPRT3, IPRT1A, IPRT2A, and IPRT3A.
N_1, N_2, N_3	The number of grid points in the ξ, η, and ζ directions.
ICHECK	Input parameter specifying frequency for checking convergence.
IPLOT	Input flag specifying type of plot file being written.
NG1, NG2, NG3	Number of points in the ξ, η, and ζ directions in the coordinate system file.

The typical record length and total size values listed in the table were computed assuming $N_{av} = 1$, $N_{turb} = 0$, $N_{eq} = 5$, $N_t = 1$, $N_{t1} = 1$, $N_{t2} = 2000$, $N_\xi = N_\eta = N_\zeta = 21$, $N_1 = N_2 = N_3 = 51$, ICHECK = 10, and NG1 = NG2 = NG3 = 51.

TABLE 6-1. - I/O FILE TYPES

UNIT	DEFAULT UNIT NO.	RECORD FORMAT	CONTENTS	WHEN USED
NIN	5	Formatted	Standard input	Always
NOUT	6	Formatted	Standard output	Always
NGRID	7	Unformatted	Coordinate system input	NGEOM = 10
NPLOTX	8	Unformatted	PLOT3D XYZ file output	IPLOT = 2, 3
NPLOT	9	Unformatted	CONTOUR plot file or PLOT3D Q file output	IPLOT $\neq$ 0
NHIST	10	Unformatted	Convergence history output	Always
NRQIN	11	Unformatted	Restart flow field input	IREST = 2, 3
NRQOUT	12	Unformatted	Restart flow field output	IREST = 1, 2, 3
NRXIN	13	Unformatted	Restart computational coordinates input	IREST = 2, 3
NRXOUT	14	Unformatted	Restart computational coordinates output	IREST = 1, 2, 3

TABLE 6-2. - I/O FILE SIZES

UNIT	MAXIMUM RECORD LENGTH[a]		TYPICAL MAXIMUM RECORD LENGTH[a]	TOTAL SIZE[b]		TYPICAL TOTAL SIZE[b]
NIN	80		80	Variable		Variable
NOUT	132		132	$\simeq (2N_{eq} + 3)$ pages + $N_\zeta\{(N_\eta + 3)[(N_\xi - 1)/10 + 1] + 2\}/55$ pages per variable per time level		13 pages + 42 pages per variable per time level
NGRID	3(NG1)(NG2)(NG3)		397,953	3(NG1)(NG2)(NG3) + 2		397,955
NPLOTX	IPLOT			IPLOT		
	2	$3N_1N_2N_3$	397,953	2	$3N_1N_2N_3 + 3$	397,956
	3	$3N_1N_2$	7,803	3	$3N_1N_2N_3 + 3$	397,956
NPLOT	IPLOT			IPLOT		
	1	$14N_3$	714	1	$(14N_1N_2N_3 + 23)N_t$	1,857,137
	2	$5N_1N_2N_3$	663,255	2	$(5N_1N_2N_3 + 7)N_t$	663,262
	3	$5N_1N_2$	13,005	3	$(5N_1N_2N_3 + 7)N_t$	663,262
NHIST	$8N_{eq}$		40	$[(N_{av} + 1)(8N_{eq}) + 3] \cdot (N_{t2} - N_{t1} + 1)/\text{ICHECK} + N_{eq} + 12$		8,617
NRQIN, NRQOUT	$5N_1N_2N_3$		663,255	$10N_1N_2N_3 + 7$		1,326,517
NRXIN, NRXOUT	$3N_1N_2N_3$		397,953	$3N_1N_2N_3 + 3$		397,956

[a] In bytes for units NIN and NOUT, and in computer words for the remaining units.

[b] In pages for units NIN and NOUT, and in computer words for the remaining units.

INTENTIONALLY BLANK

7.0 DIAGNOSTIC MESSAGES

Various diagnostic messages may be printed by *Proteus* as part of its standard output.[24] Most of these concern inconsistent or invalid input, although some describe problems encountered during the calculation itself. Two types of messages may appear - errors and warnings. Error messages are preceded by the characters `**** ERROR`, and indicate either a serious problem or an input error that *Proteus* cannot resolve. Warning messages are preceded by the characters `**** WARNING`, and indicate either a potential problem or a non-standard combination of input parameters. Errors cause the calculation to stop, while warnings do not.

The various error and warning messages are listed and explained in the following two subsections. Italic letters, like *value*, are used to indicate variable values. The subprogram in which the message is printed is given in parentheses at the end of the each explanation.

7.1 ERROR MESSAGES

`Both machr and ur specified.`
`machr =` *value*`, ur =` *value*

> Either the reference Mach number or velocity may be specified in namelist FLOW, but not both. (INPUT)

`Both prlr and ktr specified.`
`prlr =` *value*`, ktr =` *value*

> Either the reference laminar Prandtl number or thermal conductivity may be specified in namelist FLOW, but not both. (INPUT)

`Both rer and mur specified.`
`rer =` *value*`, mur =` *value*

> Either the reference Reynolds number or viscosity may be specified in namelist FLOW, but not both. (INPUT)

`Coordinate system file has ngl, ng2, and/or ng3 > max allowed.`
`ngl =` *value*`, ng2 =` *value*`, ng3 =` *value*

> A coordinate system file has been read in, using the NGEOM = 10 option, with more grid points than allowed. The maximum allowed values of NG1, NG2, and NG3 are the values of the dimensioning parameters N1P, N2P, and N3P, respectively. (GEOM)

`Grid transformation Jacobian changes sign or = 0.`

> The nonorthogonal grid transformation Jacobian J must be either everywhere positive or everywhere negative. This error indicates that the computational mesh contains crossed or coincident grid lines. The error message is followed by a printout of the Cartesian coordinates, the Jacobian, and the metric coefficients. (METS)

`Illegal option for computational coordinates requested.`
`ngeom =` *value*

> An illegal value of NGEOM has been specified in namelist GMTRY. The legal values are 1, 2, and 10, and are described in Section 3.1.5. (GEOM)

[24] The diagnostic messages described in this section are generated by the *Proteus* code itself, and appear as part of the standard output. Any computer system error messages due to floating-point errors, etc., are, of course, system-dependent. On UNIX-based systems, system errors will normally appear in the standard error file.

PRECEDING PAGE BLANK NOT FILMED

`Illegal plot file option requested.`
`iplot =` *value*

An illegal value of IPLOT has been specified in namelist IO. The legal values are 0, 1, 2, and 3, and are described in Section 3.1.4. (PLOT)

`Illegal thin-layer option.`
`ithin(`*idir*`) =` *value*

An illegal value of ITHIN(*idir*) has been specified in namelist FLOW. Here *idir* = 1, 2, or 3, corresponding to the ξ, η, and ζ directions, respectively. The legal values are 0 and 1, and are described in Section 3.1.6. (INPUT)

`Illegal time step selection option requested.`
`idtau =` *value*

An illegal value of IDTAU has been specified in namelist TIME. The legal values are 1 to 9, and are described in Section 3.1.9. (TIMSTP)

`Illegal value for icvars.`
`icvars =` *value*

An illegal value of ICVARS has been specified in namelist FLOW. The legal values are 1 to 6, and are described in Section 3.1.6. (INITC)

`Illegal value for ieuler.`
`ieuler =` *value*

An illegal value of IEULER has been specified in namelist FLOW. The legal values are 0 and 1, and are described in Section 3.1.6. (INPUT)

`Illegal value for ihstag.`
`ihstag =` *value*

An illegal value of IHSTAG has been specified in namelist FLOW. The legal values are 0 and 1, and are described in Section 3.1.6. (INPUT)

`Illegal value for ilamv.`
`ilamv =` *value*

An illegal value of ILAMV has been specified in namelist FLOW. The legal values are 0 and 1, and are described in Section 3.1.6. (FTEMP)

`Illegal value for ildamp.`
`ildamp =` *value*

An illegal value of ILDAMP has been specified in namelist TURB. The legal values are 0 and 1, and are described in Section 3.1.10. (INPUT)

`Illegal value for inner.`
`inner =` *value*

An illegal value of INNER has been specified in namelist TURB. The legal values are 1 and 2, and are described in Section 3.1.10. (INPUT)

`Illegal value for irest.`
`irest =` *value*

An illegal value of IREST has been specified in namelist RSTRT. The legal values are 0, 1, 2, and 3, and are described in Section 3.1.3. (INPUT)

`Illegal value for iunits.`
`iunits =` *value*

An illegal value of IUNITS has been specified in namelist IO. The legal values are 0 and 1, and are described in Section 3.1.4. (INPUT)

Illegal value for lwall1.
lwall1(*j,k,ibound*) = *value*

Illegal value for lwall2.
lwall2(*i,k,ibound*) = *value*

Illegal value for lwall3.
lwall3(*i,j,ibound*) = *value*

An illegal value of LWALL1, LWALL2, or LWALL3 has been specified in namelist TURB. Here *i*, *j*, and *k* are the indices in the ξ, η, and ζ directions, and *ibound* = 1 or 2, corresponding to the $\xi = 0$ or 1 surface, the $\eta = 0$ or 1 surface, or the $\zeta = 0$ or 1 surface. The legal values are 0 and 1, and are described in Section 3.1.10. (INPUT)

Improper specification of thin-layer option.
ithin = *value value value*

The thin-layer option may be used in one direction only. Thus, one element of the ITHIN array must be set equal to 1, and the remaining two must be 0. (INPUT)

Invalid boundary condition type requested.
jbc1(*ieq,ibound*) or ibc1(*j,k,ieq,ibound*) = *value*

Invalid boundary condition type requested.
jbc2(*ieq,ibound*) or ibc2(*i,k,ieq,ibound*) = *value*

Invalid boundary condition type requested.
jbc3(*ieq,ibound*) or ibc3(*i,j,ieq,ibound*) = *value*

These messages result from an invalid boundary condition type being specified in namelist BC for the ξ, η, and/or ζ direction. Here *ieq* is the boundary condition equation number; *ibound* = 1 or 2, corresponding to the $\xi = 0$ or 1 surface, the $\eta = 0$ or 1 surface, or the $\zeta = 0$ or 1 surface; and *i*, *j*, and *k* are the indices in the ξ, η, and ζ directions. The valid boundary conditions are listed in Table 3-6. (BCDENS, BCF, BCGEN, BCNVEL, BCPRES, BCQ, BCTEMP, BCUVEL, BCVVEL, BCWVEL, BC1VEL, BC2VEL, BC3VEL)

Invalid boundary type requested.
kbc1(*ibound*) = *value*

Invalid boundary type requested.
kbc2(*ibound*) = *value*

Invalid boundary type requested.
kbc3(*ibound*) = *value*

These messages result from an invalid boundary type being specified in namelist BC, for the ξ, η, and/or ζ direction, when the KBC meta flags are used. Here *ibound* = 1 or 2, corresponding to the $\xi = 0$ or 1 surface, the $\eta = 0$ or 1 surface, or the $\zeta = 0$ or 1 surface. The valid boundary types are listed in Section 3.1.7. (BCSET)

Invalid grid packing location for Roberts formula.
sq(*idir*,1) = *value*

An invalid grid packing location, given by the value of SQ(IDIR,1) in namelist NUM, has been specified. Here *idir* = 1, 2, or 3, corresponding to the ξ, η, and ζ directions, respectively. The valid values are 0.0, 0.5, and 1.0, and are described in Section 3.1.8. (INPUT)

Invalid grid packing option.
ipack(*idir*) = *value*

An invalid grid packing option, given by the value of IPACK(IDIR) in namelist NUM, has been specified. Here *idir* = 1, 2, or 3, corresponding to the ξ, η, and ζ directions, respectively. The valid values are 0 and 1, and are described in Section 3.1.8. (PAK)

```
Invalid grid packing parameter for Roberts formula.
sq(idir,2) = value
```

An invalid grid packing parameter, given by the value of SQ(*idir*,2) in namelist NUM, has been specified. Here *idir* = 1, 2, or 3, corresponding to the ξ, η, and ζ directions, respectively. The valid values are > 1, and are described in Section 3.1.8. (INPUT)

```
Invalid time step selection method for time step sequencing option.
idtau = value, ntseq = value
```

A time step selection option that adjusts $\Delta\tau$ as the solution proceeds has been specified in namelist TIME in conjunction with the time step sequencing option. If the time step sequencing option is being used, IDTAU must be 1, 3, or 5. (INPUT)

```
Invalid type of artificial viscosity requested.
iav4e, iav2e, iav2i = value value value
```

An invalid type of artificial viscosity has been specified in namelist NUM. The valid values are 0, 1, and 2 for IAV4E and IAV2E, and 0 and 1 for IAV2I, and are described in Section 3.1.8. (INPUT)

```
Invalid type of unsteadiness for boundary condition requested.
jtbc1(ieq,ibound) = value
```

```
Invalid type of unsteadiness for boundary condition requested.
jtbc2(ieq,ibound) = value
```

```
Invalid type of unsteadiness for boundary condition requested.
jtbc3(ieq,ibound) = value
```

These messages result from an invalid type of unsteadiness being specified in namelist BC for the boundary conditions in the ξ, η, and/or ζ direction. Here *ieq* is the boundary condition equation number, and *ibound* = 1 or 2, corresponding to the $\xi = 0$ or 1 surface, the $\eta = 0$ or 1 surface, or the $\zeta = 0$ or 1 surface. The valid values for JTBC1, JTBC2, and JTBC3 are 0, 1, and 2, and are described in Section 3.1.7. (TBC)

```
Mesh size requested > max allowed.
n1 = value, n2 = value, n3 = value
```

More grid points have been requested in namelist NUM than are allowed. For non-periodic boundary conditions, the maximum allowed values of N1, N2, and N3 are the values of the dimensioning parameters N1P, N2P, and N3P, respectively. For spatially periodic boundary conditions, the maximum values are N1P − 1, N2P − 1, and N3P − 1. (INPUT)

```
More time step sequences requested than allowed.
ntseq = value
```

For the time step sequencing option, the number of time step sequences, specified in namelist TIME, cannot exceed the value of the dimensioning parameter NTSEQP. (INPUT)

```
neqp not large enough.
neqp, ihstag = value value
```

The value of the dimensioning parameter NEQP, which sets the maximum number of coupled equations that can be solved, is not large enough. The number of equations to be solved depends on the value of the input parameter IHSTAG. See Section 6.2 and Table 3-4 in Section 3.0. (INPUT)

```
Non-existent turbulence model requested.
iturb = value
```

A non-zero value for ITURB has been specified in namelist FLOW that does not correspond to one of the turbulence models currently available in *Proteus*. The only valid non-zero values for ITURB are 1 and 20, corresponding to the algebraic Baldwin-Lomax and the Chien two-equation turbulence models, respectively. (INITC, MAIN)

```
Non-positive pressure and/or temperature at time level n
```

```
 I1      I2      I3      P      T
value   value   value   value  value
```

During the solution, a non-positive value for pressure and/or temperature has been computed in subroutine EQSTAT. Up to 50 values will be printed. These values, of course, are non-physical and indicate a failure of the solution. Although the calculation will stop, the standard output and plot file will include this time level, if that is consistent with the "IPRT" and "IPLT" type parameters in namelist IO. The restart files will not be written. This failure may be caused by bad initial or boundary conditions, or by too large a time step. (INITC, MAIN)

```
Number of values in unsteady boundary condition table > max allowed.
ntbc = value
```

For unsteady boundary conditions, the number of values in the tables of GTBC1, GTBC2, and/or GTBC3 vs. NTBCA, specified in namelist TIME, cannot exceed the value of the dimensioning parameter NTP. (INPUT)

```
Singular block matrix for b. c. at lower boundary, sweep n
```

```
Singular block matrix for b. c. at upper boundary, sweep n
```

When boundary conditions are specified using the JBC and/or IBC input parameters, zero values may appear on the diagonal of the block tridiagonal coefficient matrix. Subroutine FILTER attempts to rearrange the elements of the boundary condition block submatrices to eliminate any of these zero values. These messages indicate it was unable to do so for the boundary and sweep indicated. This means the diagonal submatrix **B** is singular, which in turn means the specified boundary conditions are not independent of one another. (FILTER)

7.2 WARNING MESSAGES

```
chgmax > 0.15, cfl cut in half.
```

```
chgmax > 0.15, dtau cut in half.
```

With the IDTAU = 2, 4, and 6 options, the time step is adjusted gradually as the solution proceeds based on the absolute value of the maximum change in the dependent variables. One of these messages may occur if the solution changes very rapidly. (The first message applies to the IDTAU = 2 and 6 options, and the second to the IDTAU = 4 option.) Under these conditions the time step is arbitrarily cut in half, and the solution continues. This may stabilize the calculation, but consideration should be given to rerunning the problem with a smaller time step, especially for unsteady flows. (TIMSTP)

```
Conflicting boundary conditions specified.
kbc1(ibound) = value, jbc1(ieq,ibound) = value, ibc1(j,k,ieq,ibound) = value
kbc1(ibound) = value will be used.

Conflicting boundary conditions specified.
kbc2(ibound) = value, jbc2(ieq,ibound) = value, ibc2(i,k,ieq,ibound) = value
kbc2(ibound) = value will be used.

Conflicting boundary conditions specified.
kbc3(ibound) = value, jbc3(ieq,ibound) = value, ibc3(i,j,ieq,ibound) = value
kbc3(ibound) = value will be used.
```

These messages indicate that a boundary type was specified using the KBC parameter, and that individual boundary conditions were also specified for that boundary using the JBC and/or IBC parameters, for one of the ξ, η, and/or ζ boundaries. For a given boundary, the boundary type may be specified using the KBC parameter, or individual boundary conditions may be specified using the JBC and/or IBC parameters. Here *ibound* = 1 or 2, corresponding to the $\xi = 0$ or 1 surface, the $\eta = 0$ or 1 surface, or the $\zeta = 0$ or 1 surface; *ieq* is the boundary condition equation number; and *i*, *j*, and *k* are the indices in the ξ, η, and ζ directions. Boundary conditions will be set using the KBC parameter, and the calculation will continue. See the discussion of boundary condition input in Section 3.1.7. (INPUT)

```
Conflicting boundary conditions specified.
jbc1(ieq,ibound) = value, ibc1(j,k,ieq,ibound) = value
jbc1(ieq,ibound) = value will be used.

Conflicting boundary conditions specified.
jbc2(ieq,ibound) = value, ibc2(i,k,ieq,ibound) = value
jbc2(ieq,ibound) = value will be used.

Conflicting boundary conditions specified.
jbc3(ieq,ibound) = value, ibc3(i,j,ieq,ibound) = value
jbc3(ieq,ibound) = value will be used.
```

These messages indicate that both surface and point-by-point boundary conditions were specified for the same boundary condition equation number for one of the ξ, η, and/or ζ boundaries. Each boundary condition on each boundary may be specified for the entire surface using the JBC and GBC parameters, or on a point-by-point basis using the IBC and FBC ₁ırameters, but not both. Here *ieq* is the boundary condition equation number; *ibound* = 1 or 2, corresponding to the $\xi = 0$ or 1 surface, the $\eta = 0$ or 1 surface, or the $\zeta = 0$ or 1 surface; and *i*, *j*, and *k* are the indices in the ξ, η, and ζ directions. A likely cause of this error message is specifying point-by-point boundary conditions without setting the appropriate JBC parameter equal to -1. The boundary condition will be set using the JBC parameter, and the calculation will continue. See the discussion of boundary condition input in Section 3.1.7. (INPUT)

```
icvars reset to 2 for default init.
```

The default version of subroutine INIT is set up assuming ICVARS = 2. If another value of ICVARS is read in, it is automatically reset to 2 and the calculation will continue. (INIT)

```
Illegal convergence testing option requested.
ictest = value, reset to ictest = 3
```

An illegal value of ICTEST has been specified in namelist TIME. ICTEST will be reset to 3, corresponding to convergence based on the L_2 norm of the residual, and the calculation will continue. The legal values are 1 to 5, and are described in Section 3.1.9. (CONV)

```
Illegal output requested.
ivout(n) = value
```

An illegal value of IVOUT has been specified in namelist IO. It will be ignored and the calculation will continue. The legal values of IVOUT are listed in Table 3-3. (OUTPUT)

```
Implicit & nonlinear explicit artificial viscosity both requested.
iav2i, iav2e, iav4e = value value value
cavs2i = value value value value value
```

Normally, Jameson's nonlinear artificial viscosity, specified by setting IAV2E and IAV4E = 2, is explicit only. This message is printed when implicit artificial viscosity is requested at the same time. *Proteus* will assume that you know what you are doing and the implicit artificial viscosity will be included. (INPUT)

```
Invalid debug option specified.
idebug(i)
```

An invalid debug option, number *i*, has been specified in namelist IO. The valid IDEBUG options are 1 to 7, and are described in Section 3.1.4. (INPUT)

```
Non-standard time difference centering requested.
thc = value value
thx = value value value
thy = value value value
thz = value value value
the = value value value
```

The Beam-Warming type time differencing used in *Proteus* includes three standard implicit schemes - Euler, trapezoidal, and three-point backward. This message indicates that a combination of time differencing parameters θ_1, θ_2, and θ_3 has been specified for at least one of the governing equations that does not correspond to any of the three standard schemes. *Proteus* will assume that you know what you are doing and the specified time differencing parameters will be used. (INPUT)

```
Spatially varying time step requested with time-accurate differencing scheme.
idtau = value
thc   = value value
thx   = value value value
thy   = value value value
thz   = value value value
the   = value value value
```

For steady flows, a spatially varying time step may be used to enhan. : convergence, and first-order time differencing is recommended. Using a second-order time-accurate differencing scheme should not give wrong results, but is wasteful. For unsteady flows, second-order time-accurate differencing should be used, but only a global (i.e., constant in space) time step makes sense. (INPUT)

```
Time level may fall outside range of input table for unsteady b. c.
itbeg = value, itend = value, ntbca(1) = value, ntbca(ntbc) = value
```

General unsteady boundary conditions are being used, and the time levels in the input table of GTBC1, GTBC2, and/or GTBC3 vs. NTBCA do not cover the time levels that will occur during the time marching loop. Here `itbeg` and `itend` are the first and last time levels in the time marching loop, and *ntbc* is the value of the input parameter NTBC. If the time level $n <$ NTBCA(1), the boundary condition value will be set equal to the first value in the table. Similarly, if $n >$ NTBCA(NTBC), the value will be set equal to the last value in the table. (TBC)

7.3 OTHER MESSAGES

```
Converged solution at time level n
```

The solution has converged, according to the criteria specified by the parameters ICTEST and EPS, at the indicated time level. The calculation will stop, and the standard output and plot file will include this time level, if that is consistent with the "IPRT" and "IPLT" parameters in namelist IO. The restart files will be written. (MAIN)

`Not yet converged.`

The solution has not converged, according to the criteria specified by the parameters ICTEST and EPS, within the number of time steps specified by NTIME. The calculation will stop, and the standard output and plot file will include this time level, if that is consistent with the "IPRT" and "IPLT" parameters in namelist IO. The restart files will be written. (MAIN)

`Sorry, ran out of cpu time at time level` *n*

The calculation was stopped at the indicated time level because the job was almost out of CPU time. The standard output and plot file will include this time level, if that is consistent with the "IPRT" and "IPLT" parameters in namelist IO. The restart files will be written. (MAIN)

8.0 JOB CONTROL LANGUAGE

At NASA Lewis, *Proteus* is currently being run on the Cray X-MP and Y-MP computers, with UNICOS 6.0 as the operating system. In this section several general examples are presented showing the UNICOS job control language (JCL) that may be used as starting points when setting up specific cases.[25] The individual UNICOS commands are described in detail in the *UNICOS User Commands Reference Manual,* (Cray Research, Inc., 1991). These examples are written for the Bourne shell. Some changes may be necessary if the C shell is being used. It is assumed that the user is familiar with the procedures used at their site to submit jobs to the Cray, and to receive and process the output files.

Each example is given with reference line numbers, which are not part of the actual JCL statement, followed by a line-by-line explanation. Note that in UNICOS, the case (upper or lower) of the letters in commands and arguments is significant. In this respect, the examples should be followed exactly.

8.1 COMPILING THE CODE

In this example, the *Proteus* code is compiled on the Cray. It is assumed that the source code is in the proper format for the Cray UPDATE facility, as described in the *UPDATE Reference Manual* (Cray Research, Inc., 1988).

```
1.    # QSUB -lM 2.0mW -lT 300
2.    # QSUB -mb -me
3.    # QSUB -o temp.out
4.    # QSUB -eo
5.    # QSUB -r EXAMPLE
6.    # QSUB -s /bin/sh
7.    set -x
8.    update -i p3d10.s -n p3d10.u -c p3d10 -f
9.    cft77 -b p3d10.o -d pq -o aggress p3d10.f
10.   rm p3d10.f
```

Lines 1 through 6 are actually Network Queueing System (NQS) commands, not UNICOS commands. They must appear first in the runstream, and begin with a # sign. This first line sets the memory and CPU time limits for the job at 2.0 million words and 300 seconds.

Line 2 tells the Cray to send email when the job starts and finishes.

Line 3 tells the Cray to store the standard output in the file *temp.out.*

Line 4 causes any system error messages to be included in the standard output file.

Line 5 gives the name of the job as EXAMPLE.

Line 6 causes the Bourne shell to be used for this job.

Line 7 causes your UNICOS commands to be printed as part of the output runstream.

Line 8 uses the Cray UPDATE facility to create the files, *p3d10.f,* which contains the complete compilable Fortran code for *Proteus,* and *p3d10.u,* which contains the *Proteus* update program library. The update command uses as input the file *p3d10.s.*

Line 9 compiles the program *p3d10.f,* storing the object code in the file *p3d10.o.*

Line 10 removes the named file.

[25] See Section 9.0 for specific examples of actual cases.

8.2 RUNNING THE MASTER FILE

The simplest way to run *Proteus* is shown in this example. The existing master file is being used, without making any changes.

```
1.    # QSUB -lM 4.0mW -lT 3600
2.    # QSUB -mb -me
3.    # QSUB -o temp.out
4.    # QSUB -eo
5.    # QSUB -r EXAMPLE
6.    # QSUB -s /bin/sh
7.    set -x
8.    ja
9.    touch plotx
10.   touch plotq
11.   touch chist
12.   touch rq2
13.   touch rx2
14.   ln plotx  fort.8
15.   ln plotq  fort.9
16.   ln chist  fort.10
17.   ln rq2    fort.12
18.   ln rx2    fort.14
19.   ln coords fort.7
20.   segldr -o p3d10.ex p3d10.o
21.   p3d10.ex << EOD
```

Standard Proteus input file goes here

```
22.   EOD
23.   rm p3d10.ex
24.   ja -cslt
```

Lines 1 through 6 are actually Network Queueing System (NQS) commands, not UNICOS commands. They must appear first in the file, and begin with a # sign. This first line sets the memory and CPU time limits for the job at 4.0 million words and 3600 seconds.

Line 2 tells the Cray to send email when the job starts and finishes.

Line 3 tells the Cray to store the standard output in the file *temp.out*.

Line 4 causes any system error messages to be included in the standard output file.

Line 5 gives the name of the job as EXAMPLE.

Line 6 causes the Bourne shell to be used for this job.

Line 7 causes your UNICOS commands to be printed as part of the output runstream.

Line 8 tells the Cray to begin keeping accounting information for later printing.

Lines 9-13 create empty files (assuming they don't already exist) with the file names as shown.

Lines 14-18 link these files with the indicated Fortran unit numbers. The files are thus the PLOT3D XYZ file, the PLOT3D Q file or CONTOUR plot file, the convergence history file, and the output restart flow field and mesh files. These lines implicitly open the files for input and output. Fortran OPEN statements are not used in *Proteus*. If the *Proteus* input is such that any of these files are unnecessary (see Table 6-1), then the *touch* and *ln* for those files can be eliminated.

Line 19 links an existing computational coordinate system file with Fortran unit 7. If the input parameter NGEOM $\neq$ 10, this line should be eliminated. If a restart case is being run (IREST = 2 or 3), this line should be replaced by the following two lines:

```
ln rq1 fort.11
ln rx1 fort.13
```

The above lines link existing input restart flow field and computational mesh files with Fortran units 11 and 13.

Line 20 creates an executable file, *p3d10.ex*, from the existing object file *p3d10.o*.

Line 21 actually runs the program. The standard *Proteus* input consists of all the records between line 21 and line 22, which contains the marker "EOD". The input could also be obtained from a file on the Cray. In that case, line 21 should be replaced by:

```
p3d10.ex < input
```

where *input* is the name of the file, and line 22 should be eliminated.

Line 23 removes the named file.

Line 24 causes various accounting information to be printed at the end of your output.

8.3 MODIFYING THE MASTER FILE, FULL UPDATE

This example shows how to run with a temporarily modified version of the master file. In this particular case, the existing master file is modified to increase the dimensioning parameters N1P, N2P, and N3P, thus allowing more mesh points to be used. Since this affects almost every subroutine, the entire program is updated and recompiled.

```
1.    # QSUB -lM 4.0mW -lT 3600
2.    # QSUB -mb -me
3.    # QSUB -o temp.out
4.    # QSUB -eo
5.    # QSUB -r EXAMPLE
6.    # QSUB -s /bin/sh
7.    set -x
8.    ja
9.    touch plotx
10.   touch plotq
11.   touch chist
12.   touch rq2
13.   touch rx2
14.   ln plotx  fort.8
15.   ln plotq  fort.9
16.   ln chist  fort.10
17.   ln rq2    fort.12
18.   ln rx2    fort.14
19.   ln coords fort.7
20.   cat > mods << EOF
      *id temp
      *d params1.20
            parameter (n1p = 51, n2p = 51, n3p = 51)
21.   EOF
22.   update -p p3d10.u -i mods -c temp -f
23.   cft77 -d pq -o aggress temp.f
24.   segldr -o temp.ex temp.o
25.   temp.ex << EOD

      Standard Proteus input file goes here

26.   EOD
27.   rm mods temp.f temp.o temp.ex
28.   ja -cslt
```

Lines 1 through 19 are the same as in the example in Section 8.2.

Line 20 creates a Cray file called *mods*. The file will consist of all the records between line 20 and line 21, which contains the marker "EOF". Your Cray UPDATE directives and new code should therefore be inserted between lines 20 and 21. The UPDATE directives and new code could also be obtained from a file on the Cray. In that case, lines 20 and 21 should be eliminated.

Line 22 uses the Cray UPDATE facility to create a file called *temp.f*, which contains the complete Fortran code for the modified version of *Proteus*. The update command uses as input the existing *Proteus* program library *p3d10.u*, and the file *mods* containing the UPDATE directives and new code.

Line 23 compiles the modified program *temp.f*, storing the object code in the file *temp.o*.

Line 24 creates an executable file, *temp.ex,* from the object file *temp.o.*

Line 25 actually runs the program. The standard *Proteus* input consists of all the records between line 25 and line 26, which contains the marker "EOD". The input could also be obtained from a file on the Cray. In that case, line 25 should be replaced by:

```
temp.ex < input
```

where *input* is the name of the file, and line 26 should be eliminated.

Line 27 removes the named files.

Line 28 causes various accounting information to be printed at the end of your output.

8.4 MODIFYING THE MASTER FILE, PARTIAL UPDATE

This example shows how to run with temporarily modified versions of just a few routines. In this particular case, the default version of subroutine INIT is replaced by a user-supplied version, and an additional user-supplied geometry option is added to subroutine GEOM. Since these changes affect only INIT and GEOM, only these subroutines are updated and recompiled.

```
1.     # QSUB -lM 4.0mW -lT 3600
2.     # QSUB -mb -me
3.     # QSUB -o temp.out
4.     # QSUB -eo
5.     # QSUB -r EXAMPLE
6.     # QSUB -s /bin/sh
7.     set -x
8.     ja
9.     touch plotx
10.    touch plotq
11.    touch chist
12.    touch rq2
13.    touch rx2
14.    ln plotx  fort.8
15.    ln plotq  fort.9
16.    ln chist  fort.10
17.    ln rq2    fort.12
18.    ln rx2    fort.14
19.    ln scrl   fort.20
20.    ln coords fort.7
21.    cat > mods << EOF
       *id temp
       *purgedk init
       *deck init

       User-supplied version of INIT goes here

       *i geom.107

       New user-supplied geometry option goes here

22.    EOF
23.    update -p p3d10.u -i mods -c temp -q geom init
24.    cft77 -d pq -o aggress temp.f
25.    segldr -o temp.ex temp.o p3d10.o
26.    temp.ex << EOD

       Standard Proteus input file goes here

27.    EOD
28.    rm mods temp.f temp.o temp.ex
29.    ja -cslt
```

Lines 1 through 20 are the same as in the example in Section 8.2.

Line 21 creates a Cray file called *mods*. The file will consist of all the records between line 21 and line 22, which contains the marker "EOF". Your Cray UPDATE directives and new code should therefore be inserted between lines 21 and 22. The UPDATE directives and new code could also be obtained from a file on the Cray. In that case, lines 21 and 22 should be eliminated.

Line 23 uses the Cray UPDATE facility to create a file called *temp.f*, which contains the Fortran code for the modified versions of subroutines GEOM and INIT. The update command uses as input the existing *Proteus* program library *p3d10.u*, and the file *mods* containing the UPDATE directives and new code.

Line 24 compiles the modified versions of GEOM and INIT, contained in the file *temp.f*, storing the object code in the file *temp.o*.

Line 25 creates an executable file, *temp.ex*, from the object file *temp.o* containing the modified versions of GEOM and INIT, and from the existing object file *p3d10.o*.

Line 26 actually runs the program. The standard *Proteus* input consists of all the records between line 26 and line 27, which contains the marker "EOD". The input could also be obtained from a file on the Cray. In that case, line 27 should be replaced by:

```
temp.ex < input
```

where *input* is the name of the file, and line 27 should be eliminated.

Line 28 removes the named files.

Line 29 causes various accounting information to be printed at the end of your output.

PAGE ____ INTENTIONALLY BLANK

9.0 TEST CASES

In this section, two test cases are described in detail. The discussion of each test case includes a description of the problem, listings of the *Proteus* input file and the JCL, and figures illustrating the computed results. Both cases were run using version 1.0 of the 3-D *Proteus* code on the NASA Lewis Cray Y-MP, running UNICOS 6.0, and Release 5.0.2 of CFT77.

9.1 DEVELOPING FLOW IN A RECTANGULAR DUCT

Problem Description

In this test case, laminar developing flow was computed in the entrance region of a straight rectangular duct. The ratio of the cross-section width to height was 5:1. A schematic of part of the duct is shown in Figure 9.1. This flow was studied experimentally by Sparrow, Hixon, and Shavit (1967).

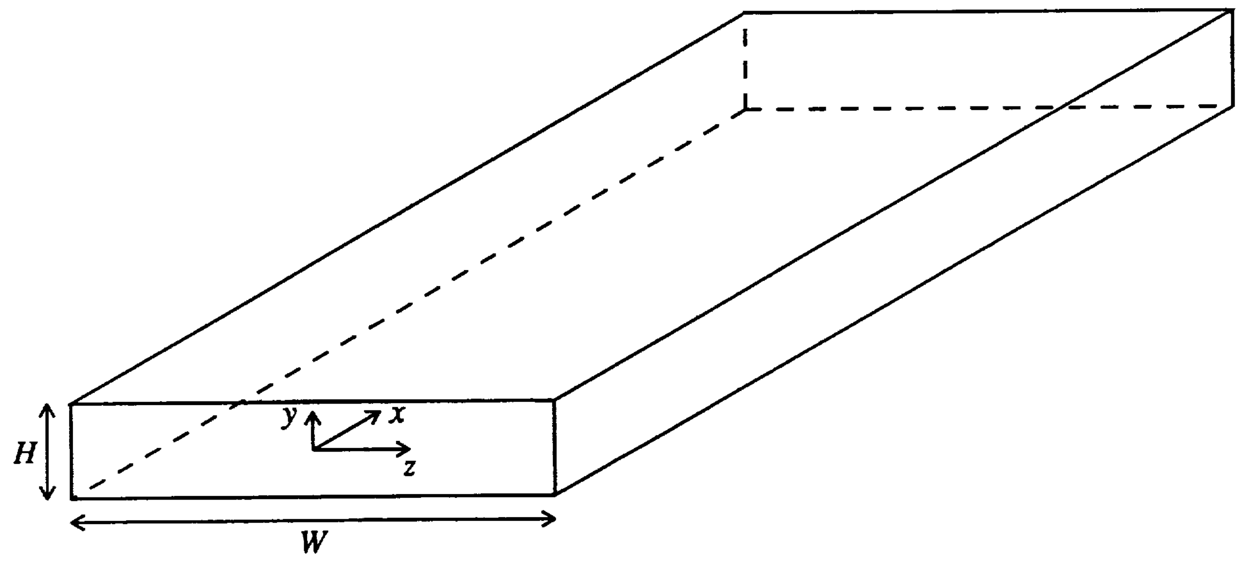

Figure 9.1. Rectangular duct.

Reference Conditions

A convenient choice for the reference length L_r was the cross-section height H, which was set equal to 1.0 ft. The default standard sea level conditions for air of 519 °R and 0.07645 lb_m/ft^3 were used for the reference temperature and density. The specific heat ratio γ_r was set to 1.4. Since the experiment was incompressible, the reference Mach number M_r was set equal to 0.1 to minimize compressibility effects. In order to reach steady state within a relatively small number of time steps, the reference Reynolds number Re_r was set equal to 60. This corresponds to $Re_D = 100$, where $D = 1.6667$ ft. is the hydraulic diameter.

Computational Coordinates

For this problem a simple Cartesian computational coordinate system can be used. The physical (x-y-z) and computational (ξ-η-ζ) coordinates are thus in the same directions. Because the flow is symmetric about both the y and z axes, the computational domain was just one quadrant of the cross-section.

Since $L_r = H$, the coordinate limits in the y and z directions were $y_{min} = 0$, $y_{max} = 0.5$, $z_{min} = 0$, and $z_{max} = 2.5$, respectively. The experimental data indicated that the flow should become fully developed within 10 hydraulic diameters. The coordinate limits in the x direction were thus $x_{min} = 0$ and $x_{max} = 16.667$.

Initial Conditions

Constant stagnation enthalpy was assumed, so only four initial conditions were required. They were simply $u = 1$, $v = w = 0$, and $p = 1$ everywhere in the flow field.

Boundary Conditions

Similarly, only four boundary conditions were required at each computational boundary. At the duct inlet the total pressure was specified, the velocity u was extrapolated, and the gradients of the velocities v and w were set equal to zero. The inlet total pressure was calculated from the freestream static pressure and the reference Mach number using isentropic relations. At the duct exit, the static pressure was specified, and the velocities u, v, and w were extrapolated. The exit static pressure was found by trial and error in order to closely match the experimental mass flow rate and the average velocity. Since *Proteus* is a compressible code, the density will change slightly, causing the average velocity to vary with x. For the exit static pressure finally used, the average velocity varied from 0.973 at the inlet to 1.039 at the exit. At the walls of the duct no-slip conditions were used for the velocities, and the normal pressure gradient was set to zero. Symmetry conditions were used in both symmetry planes. These boundary conditions are summarized in the following table.

Boundary	Boundary Conditions
$\xi = 0$	u extrapolated, $\partial v/\partial \xi = \partial w/\partial \xi = 0$, $p_T = 1.007$
$\xi = 1$	u, v, and w extrapolated, $p = 0.937$
$\eta = 0$	$\partial u/\partial \eta = 0$, $v = 0$, $\partial w/\partial \eta = \partial p/\partial \eta = 0$
$\eta = 1$	$u = v = w = 0$, $\partial p/\partial \eta = 0$
$\zeta = 0$	$\partial u/\partial \zeta = 0$, $\partial v/\partial \zeta = 0$, $w = 0$, $\partial p/\partial \zeta = 0$
$\zeta = 1$	$u = v = w = 0$, $\partial p/\partial \zeta = 0$

Proteus JCL and Input File

The Cray UNICOS job control language used for this case is listed below, including the namelist input. Note that since the default for IVOUT(5) and IVOUT(6) are 30 and 40, they are set to zero in namelist IO to avoid printout of the static pressure and temperature. The ξ indices specified for the printout correspond to the locations where velocity profiles were measured in the experiment. In namelist FLOW, the only specified reference conditions are MACHR and RER, since the desired values for the remaining ones are the same as the default values. ILAMV is defaulted, resulting in constant viscosity μ and thermal conductivity k. In namelist BC, the JBC values corresponding to derivative boundary conditions are positive, specifying that two-point one-sided differences are to be used. In namelist NUM, the parameters IPACK and SQ for the ξ and η directions are defaulted, which results in an evenly spaced mesh in those directions. The artificial viscosity is turned off because of the low Reynolds number.

The input includes a brief explanation of each line, and should be compared with the detailed input description in Section 3.0. Note that it is assumed that *Proteus* has already been compiled, using the procedure described in Section 8.0, with the dimensioning parameters N1P, N2P, and N3P equal to 101, 21, and 41, respectively.

```
# QSUB -lM 8.0mW -lT 7200
# QSUB -mb -me
# QSUB -o temp.out
# QSUB -eo
# QSUB -r duct5
# QSUB -s /bin/sh
set -x
ja

#  Set up and link necessary input/output files
```

```
touch plotx
touch plotq
touch chist
ln plotx fort.8
ln plotq fort.9
ln chist fort.10

#  Load and run Proteus

segldr -o p3d10.ex p3d10.o
p3d10.ex << EOD
Laminar developing flow in a rectangular duct
 &rstrt                                 Not a restart case.
 &end
 &io
  ivout=1,2,3,32,46*0,                  Print u, v, w, c_p.
  iprt=10000,                           Print first and last time levels only.
  iprtla=2,3,5,6,8,10,13,15,19,26,40,47,63,91,   ξ indices for printout.
  iprt2=1, iprt3=2,                     Print at all η indices, every other ζ index.
  iplot=2,                              Write PLOT3D/WHOLE plot files.
 &end
 &gmtry
  ngeom=1,                              Cartesian computational coordinates.
  xmin=0.0, xmax=16.667,                x-coordinate limits.
  ymin=0.0, ymax=0.5,                   y-coordinate limits.
  zmin=0.0, zmax=2.5,                   z-coordinate limits.
 &end
 &flow
  ihstag=1,                             Constant stagnation enthalpy.
  machr=0.1, rer=60.,                   Set M_r and Re_r.
  gamr=1.4,                             Constant specific heats.
 &end
 &bc
  jbc1(1,1)=47, jbc1(1,2)=41, gbc1(1,1)=1.007, gbc1(1,2)=0.937,
                                        p_T, p specified at ξ = 0, 1.
  jbc1(2,1)=14, jbc1(2,2)=14,           u extrapolated at ξ = 0, 1.
  jbc1(3,1)=22, jbc1(3,2)=24,            ∂v/∂ξ = 0, extrapolated at ξ = 0, 1.
  jbc1(4,1)=32, jbc1(4,2)=34,            ∂w/∂ξ = 0, extrapolated at ξ = 0, 1.
  jbc2(1,1)=42, jbc2(1,2)=42,           ∂p/∂η = 0 at η = 0, 1.
  jbc2(2,1)=12, jbc2(2,2)=11,           ∂u/∂η = 0, u = 0 at η = 0, 1.
  jbc2(3,1)=21, jbc2(3,2)=21,            v = 0 at η = 0, 1.
  jbc2(4,1)=32, jbc2(4,2)=31,            ∂w/∂η = 0, w = 0 at η = 0, 1.
  jbc3(1,1)=42, jbc3(1,2)=42,            ∂p/∂ζ = 0 at ζ = 0, 1.
  jbc3(2,1)=12, jbc3(2,2)=11,            ∂u/∂ζ = 0, u = 0 at ζ = 0, 1.
  jbc3(3,1)=22, jbc3(3,2)=21,            ∂v/∂ζ = 0, v = 0 at ζ = 0, 1.
  jbc3(4,1)=31, jbc3(4,2)=31,            w = 0 at ζ = 0, 1.
 &end
 &num
  n1=101, n2=21, n3=41,                 Use a 101 × 21 × 41 mesh.
  iav4e=0, iav2e=0, iav2i=0,            No artificial viscosity.
  ipack(3)=1,                           Pack in the ζ direction.
  sq(3,1)=1., sq(3,2)=1.1,              Pack moderately near ζ = 1.
 &end
 &time
  idtmod=10,                            Recompute Δτ every 10 steps.
  idtau=5, cfl=10.,                     Spatially varying Δτ.
  ntime=1500,                           Limit of 1,500 time steps.
 &end
 &turb                                  Laminar flow.
 &end
 &ic
  u0=1.,                                Use u = p = 1, v = w = 0 as initial conditions.
 &end
EOD
```

```
ja -cslt
```

Computed Results

For these calculations, the convergence criterion was that the L_2 norm of the residual for each equation be less than 10^{-3}. After 1,500 time steps, the solution had not yet converged to this level, although the residuals were continuing to drop. The largest L_2 norm, which was for the x-momentum equation, had dropped from about 10^4 to 10^{-1}. The maximum and average residuals for the x-momentum equation had dropped from about 4×10^2 to 2×10^{-3}, and 2×10^1 to 3×10^{-4}, respectively. Continuing the calculation past 1500 time steps resulted in further decreases in the residuals, but did not change the solution appreciably, so the calculation was stopped. The 1,500 iterations required 5,797 seconds of CPU time.

In Figure 9.2, the computed static pressure coefficient is compared with the experimental results of Sparrow, Hixon, and Shavit (1967). In this figure the static pressure coefficient is defined as $(p_0 - p)/q$, where p_0 is the inlet total pressure, p is the local static pressure, and $q = \rho u_{avg}^2/2$ is the dynamic pressure. The abscissa is $x/(DRe_D) \times 10^2$, where x is the axial distance, D is the hydraulic diameter, and Re_D is the Reynolds number based on the average velocity and hydraulic diameter. The computed values were generated from the *Proteus* results assuming $p_0 = p_{inlet} + q$ and $\rho = u_{avg} = 1$. The centerline value of the static pressure was used for p. The agreement between the computed and experimental results is good over most of the duct length. The largest differences are near the duct inlet and exit. This could be due to the variation of the density and average velocity with x, as discussed earlier.

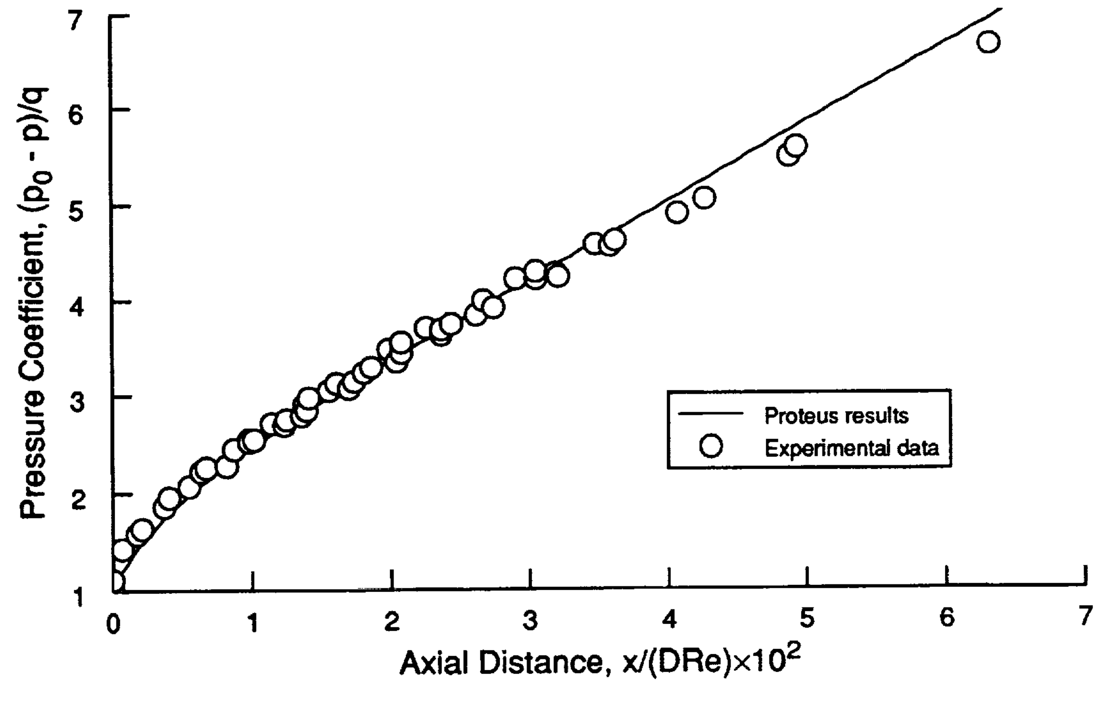

Figure 9.2. Computed static pressure distribution for rectangular duct flow.

Figures 9.3 and 9.4 show the streamwise velocity profiles as functions of y and z, respectively, at various x stations. The computed profiles are actually at the coordinate line in the computational grid that is nearest to the x station used in the experiment. No interpolation in x was done. The computed values were generated from the *Proteus* results assuming $u_{avg} = 1$. The agreement between the computational and experimental results is very good.

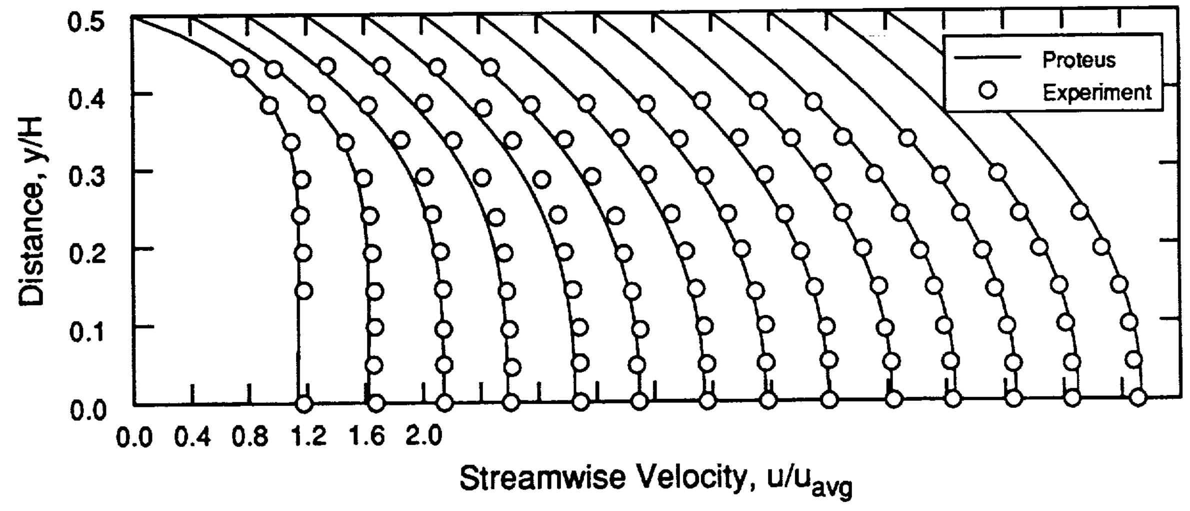

Figure 9.3. Computed velocity profiles in *x-y* plane for rectangular duct flow.

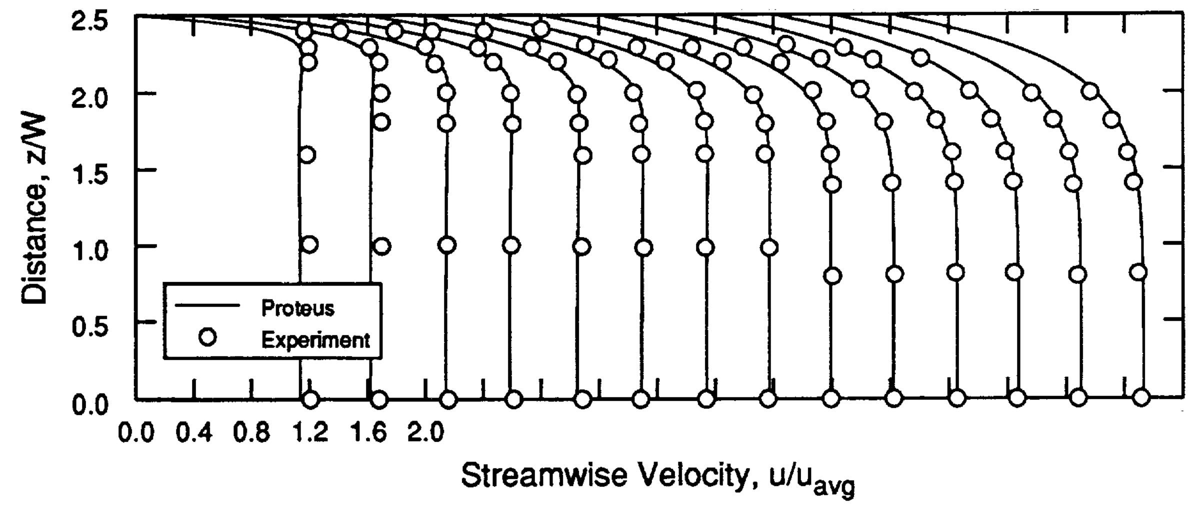

Figure 9.4. Computed velocity profiles in *x-z* plane for rectangular duct flow.

9.2 TURBULENT S-DUCT FLOW

Problem Description

In this test case, turbulent flow in an S-duct was computed using first the Baldwin-Lomax algebraic turbulence model and then the Chien k-ε turbulence model. The S-duct consisted of two 22.5° bends with a constant area square cross section. The geometry and experimental data were obtained from a test conducted by Taylor, Whitelaw, and Yianneskis (1982).

Reference Conditions

The default standard sea level conditions for air of 519 °R and 0.07645 lb_m/ft^3 were used for the reference temperature and density. The specific heat ratio γ_r was set to 1.4. Since the experiment was incompressible, the reference Mach number M_r was set equal to 0.2 to minimize compressibility effects and, at the same time, achieve a reasonable convergence rate with the *Proteus* code. In the experiment, the Reynolds number based on the bulk velocity and the hydraulic diameter was 40,000. This value was therefore used as the reference Reynolds number Re_r in the calculation. The reference length L_r was set equal to 0.028658 ft. This value was computed from the definition of Re_r, where M_r and Sutherland's law were used to compute u_r and μ_r, respectively.

Computational Coordinates

Figure 9.5 shows the computational grid for the S-duct, created using the GRIDGEN codes (Steinbrenner, Chawner, and Fouts, 1991). For clarity, the grid is shown only on three of the computational boundaries, and the points have been thinned by a factor of two in each direction. The boundary grids were first created using the GRIDGEN 2D program. The 3-D volumetric grid was then generated from the boundary grids using GRIDGEN 3D.[26]

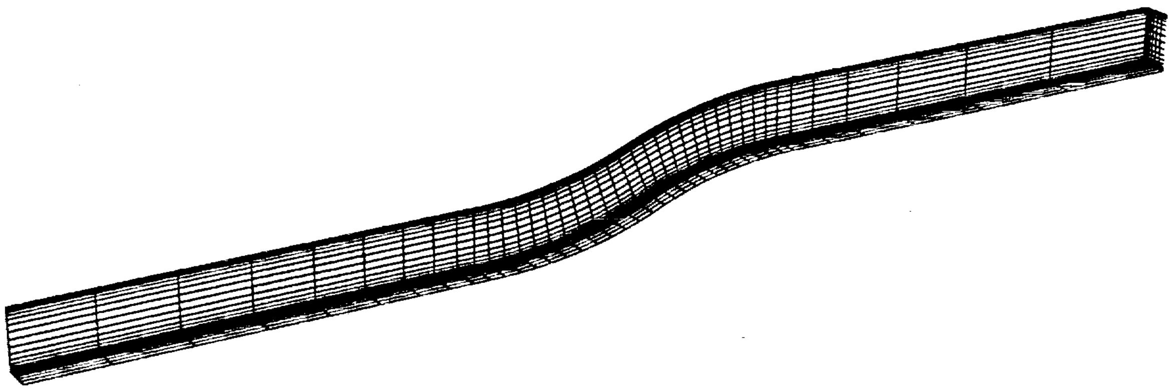

Figure 9.5. S-duct computational grid.

The computational grid extended from 7.5 hydraulic diameters upstream of the start of the first bend, to 7.5 hydraulic diameters downstream of the end of the second bend. The grid consisted of 81 x 31 x 61 points in the ξ, η, and ζ directions, respectively. Since the S-duct is symmetric with respect to the $\eta = 1$ plane, only half of the duct was computed. To resolve the viscous layers, grid points were tightly packed near the solid walls using the default packing option in GRIDGEN 2D. At the grid point nearest the wall, the value of y^+ was about 0.5.

Initial Conditions

The computations were done in two separate major steps: a calculation using the Baldwin-Lomax turbulence model and a calculation using the Chien k-ε model. To start the Baldwin-Lomax calculations, the default initial profiles specified in subroutine INIT were used. Thus, the static pressure p was set equal to 1.0, and the velocity components u, v, and w were set equal to 0.0 everywhere in the duct. To start the Chien k-ε calculations, the initial values of u, v, w, p, and the turbulent viscosity μ_t were obtained from the Baldwin-Lomax solution. The initial values of k and ε were obtained using the default KEINIT subroutine in *Proteus*.

Boundary Conditions

For both calculations, constant stagnation enthalpy was assumed, eliminating the need for solving the energy equation. Therefore, only four boundary conditions were required for the mean flow at each computational boundary. In addition, for the Chien calculation, boundary conditions were required for k and ε at each computational boundary.

[26] The form of the unformatted file created by GRIDGEN 3D was modified slightly to match that required by *Proteus*, as described in Section 3.2.

For the Baldwin-Lomax calculation, at the duct inlet the total pressure was specified, the gradient of u was set equal to zero, and the velocities v and w were set equal to zero. The inlet total pressure was calculated from the freestream static pressure and the reference Mach number using isentropic relations. At the duct exit, the static pressure was specified, and the gradients of u, v, and w were set equal to zero. The exit static pressure was found by trial and error in order to match the experimental mass flow rate. At the walls of the duct no-slip conditions were used for the velocities, and the normal pressure gradient was set to zero. Symmetry conditions were used in the symmetry plane. These boundary conditions are summarized in the following table.

Boundary	Boundary Conditions
$\xi = 0$	$\partial u/\partial \xi = 0,\ v = w = 0,\ p_T = 1.028281121$
$\xi = 1$	$\partial u/\partial \xi = \partial v/\partial \xi = \partial w/\partial \xi = 0,\ p = 0.98415512$
$\eta = 0$	$u = v = w = 0,\ \partial p/\partial \eta = 0$
$\eta = 1$	$\partial u/\partial \eta = 0,\ v = 0,\ \partial w/\partial \eta = \partial p/\partial \eta = 0$
$\zeta = 0$	$u = v = w = 0,\ \partial p/\partial \zeta = 0$
$\zeta = 1$	$u = v = w = 0,\ \partial p/\partial \zeta = 0$

For the Chien calculation, the boundary conditions for the mean flow were the same as for the Baldwin-Lomax calculation, with one exception. At the duct exit, the value of the static pressure was changed slightly, from 0.98415512 to 0.98473857, again in order to match the experimental mass flow rate. For the k-ε equations, at the upstream boundary the gradients of the turbulent kinetic energy k and the turbulent dissipation rate ε were set equal to zero for the first 20 time steps. After that time, the values of k and ε were kept constant. At the downstream boundary, the gradients of k and ε were set equal to zero. No-slip conditions were used at the surface of the plate, and symmetry conditions were used at the symmetry boundary. The boundary conditions for k and ε are summarized in the following table.

Boundary	Boundary Conditions
$\xi = 0$	$\partial k/\partial \xi = \partial \varepsilon/\partial \xi = 0$ for $n \leq 21$
	$k = k^{21},\ \varepsilon = \varepsilon^{21}$ for $n > 21$
$\xi = 1$	$\partial k/\partial \xi = \partial \varepsilon/\partial \xi = 0$
$\eta = 0$	$k = \varepsilon = 0$
$\eta = 1$	$\partial k/\partial \eta = \partial \varepsilon/\partial \eta = 0$
$\zeta = 0$	$\partial k/\partial \zeta = \partial \varepsilon/\partial \zeta = 0$
$\zeta = 1$	$\partial k/\partial \zeta = \partial \varepsilon/\partial \zeta = 0$

JCL and Input for Baldwin-Lomax Calculations

The first set of calculations used the Baldwin-Lomax algebraic turbulence model. The Cray UNICOS JCL file for the run is listed below. The contents of the input section of this listing should be compared with the detailed input description in Section 3.0. Note that for the first run, the geometry file for the S-duct must be available for *Proteus* to read in. Note also that it is assumed that *Proteus* has already been compiled, using the procedure described in Section 8.0, with the dimensioning parameters N1P, N2P, and N3P equal to 81, 31, and 61, respectively.

Since the flow field is impulsively started from zero velocity everywhere, large CFL numbers specified at the very beginning of the calculation might result in an unphysical flow field and cause the calculation to blow up. Therefore, the calculations were run with CFL = 1 for the first 100 iterations, CFL = 5 for the next 200 iterations, and CFL = 10 for the remaining iterations.

```
# QSUB -lM 10.0mW -lT 14400
# QSUB -mb -me
# QSUB -o temp.out
# QSUB -eo
# QSUB -r sdturb
# QSUB -s /bin/sh
set -x
ja

# Set up and link necessary input/output files
```

```
touch plotx
touch plotq
touch chist
touch rql
touch rxl
ln sdcoords fort.7
ln plotx    fort.8
ln plotq    fort.9
ln chist    fort.10
ln rql      fort.12
ln rxl      fort.14

# Load and run Proteus

segldr -o p3d10.ex p3d10.o
p3d10.ex << EOD
Turbulent S-Duct Flow, Re = 40,000, Baldwin-Lomax, NASA CR 3550
 &rstrt
  irest=1,                            Write restart files.
 &end
 &io
  ivout=13, 49*0,
  iprt=100000,                        Print every 100,000 time levels.
  iprt1=5, iprt2a=1, iprt3=2          Print at every 5th. ξ and 2nd. ζ index,
                                           in symmetry plane only.
  iplot=2,                            Write PLOT3D plot files.
 &end
 &gmtry
  ngeom=10,                           Get computational coordinates from file.
 &end
 &flow
  ihstag=1,                           Constant stagnation enthalpy.
  lr=0.028657852,                     Set L_r.
  machr=0.2, rer=40000.,              Set M_r and Re_r.
  gamr=1.4,                           Constant specific heats.
 &end
 &bc
  jbc1(1,1)=47, jbc1(1,2)=41, gbc1(1,1)=1.028281121, gbc1(1,2)=0.98415512,
                                      p_T, p specified at ξ = 0, 1.
  jbc1(2,1)=12, jbc1(2,2)=12,          ∂u/∂ξ = 0 at ξ = 0, 1.
  jbc1(3,1)=21, jbc1(3,2)=22,          v = 0, ∂v/∂ξ = 0 at ξ = 0, 1.
  jbc1(4,1)=31, jbc1(4,2)=32,          w = 0, ∂w/∂ξ = 0 at ξ = 0, 1.
  jbc2(1,1)=42, jbc2(1,2)=42,          ∂p/∂η = 0 at η = 0, 1.
  jbc2(2,1)=11, jbc2(2,2)=12,          u = 0, ∂u/∂η = 0 at η = 0, 1.
  jbc2(3,1)=21, jbc2(3,2)=21,          v = 0 at η = 0, 1.
  jbc2(4,1)=31, jbc2(4,2)=32,          w = 0, ∂w/∂η = 0 at η = 0, 1.
  jbc3(1,1)=42, jbc3(1,2)=42,          ∂p/∂ζ = 0 at ζ = 0, 1.
  jbc3(2,1)=11, jbc3(2,2)=11,          u = 0 at ζ = 0, 1.
  jbc3(3,1)=21, jbc3(3,2)=21,          v = 0 at ζ = 0, 1.
  jbc3(4,1)=31, jbc3(4,2)=31,          w = 0 at ζ = 0, 1.
 &end
 &num
  n1=81, n2=31, n3=61,                Use an 81 x 31 x 61 mesh.
 &end
 &time
  idtmod=1,                           Recompute Δτ every time step.
  ntseq=3,                            Use 3 time step sequences.
  idtau=5, cfl=1.,5.,10.,             Spatially varying Δτ, CFL = 1, then 5, then 10.
  ntime=100,200,700,                  100 time steps in first sequence, 200 in second,
                                           700 in third.
  ictest=4, eps=4*1.0e-6,             Use R_avg = 10^-6 as convergence criteria.
 &end
 &turb
  iturb=1,                            Turbulent flow, Baldwin-Lomax model.
 &end
 &ic
 &end
EOD

ja -cslt
```

Two more runs were made with the Baldwin-Lomax model. The JCL and input for the second run were similar to that for the first run. The only differences were:

1. The *touch* and *ln* commands for the restart files were changed to:

   ```
   touch rq2
   touch rx2
   ln rq1 fort.11
   ln rx1 fort.13
   ln rq2 fort.12
   ln rx2 fort.14
   ```

2. In namelist RSTRT, IREST was set equal to 2.
3. In namelist GMTRY, NGEOM was omitted, since the computational mesh is read from the restart file.
4. In namelist TIME, NTSEQ was omitted, and the parameters NTIME and CFL were changed, as follows:

`idtau=5, cfl=10.,`	*Spatially varying* $\Delta\tau$, CFL = 10.
`ntime=1500,`	*1500 time steps.*

The JCL and input for the third run were similar to that for the second run. The only differences were that the *touch* and *ln* commands for the restart files were changed to:

```
touch rq3
touch rx3
ln rq2 fort.11
ln rx2 fort.13
ln rq3 fort.12
ln rx3 fort.14
```

JCL and Input for Chien Calculations

The second set of calculations used the Chien k-ε turbulence model. It was run in a series of steps. The first run was a restart from the Baldwin-Lomax calculation. In this run, the gradients of k and ε were set to zero at the inlet. Again, a small CFL number was used at the beginning of the calculation. The first run used CFL = 1, for just 20 iterations. The second run used CFL = 1 for the first 100 iterations, CFL = 5 for the next 500 iterations, and CFL = 10 for the remaining iterations. Subsequent runs used CFL = 10 for all iterations.

The Cray UNICOS JCL file for the first run is listed below. The contents of the input section of this listing should be compared with the detailed input description in Section 3.0.

```
# QSUB -lM 10.0mW -lT 300
# QSUB -mb -me
# QSUB -o temp.out
# QSUB -eo
# QSUB -r sdturb
# QSUB -s /bin/sh
set -x
ja

# Set up and link necessary input/output files

touch plotx
touch plotq
```

```
touch chist
touch rx4
touch rq4
ln plotx fort.8
ln plotq fort.9
ln chist fort.10
ln rq3   fort.11
ln rx3   fort.13
ln rq4   fort.12
ln rx4   fort.14

# Load and run Proteus

segldr -o p3d10.ex p3d10.o
p3d10.ex << EOD
Turbulent S-Duct Flow, Re = 40,000, Chien, NASA CR 3550
 &rstrt
  irest=3,                               Read restart files, previous run used Baldwin-Lomax.
 &end
 &io
  ivout=13, 49*0,
  iprt=100000,                           Print every 100,000 time levels.
  iprt1=5, iprt2a=1, iprt3=2             Print at every 5th. ξ and 2nd. ζ index,
                                            in symmetry plane only.
  iplot=2,                               Write PLOT3D plot files.
 &end
 &gmtry
  ngeom=10,                              Get computational coordinates from file.
 &end
 &flow
  ihstag=1,                              Constant stagnation enthalpy.
  lr=0.028657852,                        Set L_r.
  machr=0.2, rer=40000.,                 Set M_r and Re_r.
  gamr=1.4,                              Constant specific heats.
 &end
 &bc
  jbctl(1,1)= 2, jbctl(1,2)= 2,          ∂k/∂ξ = 0 at ξ = 0, 1.
  jbctl(2,1)=12, jbctl(2,2)=12,          ∂ε/∂ξ = 0 at ξ = 0, 1.
  jbct2(1,1)= 1, jbct2(1,2)= 2,          k = 0, ∂k/∂η = 0 at η = 0, 1.
  jbct2(2,1)=11, jbct2(2,2)=12,          ε = 0, ∂ε/∂η = 0 at η = 0, 1.
  jbct3(1,1)= 1, jbct3(1,2)= 1,          k = 0 at ζ = 0, 1.
  jbct3(2,1)=11, jbct3(2,2)=11,          ε = 0 at ζ = 0, 1.
  jbcl(1,1)=47, jbcl(1,2)=41, gbcl(1,1)=1.028281121, gbcl(1,2)=0.98473857,
                                         p_T, p specified at ξ = 0, 1.
  jbcl(2,1)=12, jbcl(2,2)=12,            ∂u/∂ξ = 0 at ξ = 0, 1.
  jbcl(3,1)=21, jbcl(3,2)=22,            v = 0, ∂v/∂ξ = 0 at ξ = 0, 1.
  jbcl(4,1)=31, jbcl(4,2)=32,            w = 0, ∂w/∂ξ = 0 at ξ = 0, 1.
  jbc2(1,1)=42, jbc2(1,2)=42,            ∂p/∂η = 0 at η = 0, 1.
  jbc2(2,1)=11, jbc2(2,2)=12,            u = 0, ∂u/∂η = 0 at η = 0, 1.
  jbc2(3,1)=21, jbc2(3,2)=21,            v = 0 at η = 0, 1.
  jbc2(4,1)=31, jbc2(4,2)=32,            w = 0, ∂w/∂η = 0 at η = 0, 1.
  jbc3(1,1)=42, jbc3(1,2)=42,            ∂p/∂ζ = 0 at ζ = 0, 1.
  jbc3(2,1)=11, jbc3(2,2)=11,            u = 0 at ζ = 0, 1.
  jbc3(3,1)=21, jbc3(3,2)=21,            v = 0 at ζ = 0, 1.
  jbc3(4,1)=31, jbc3(4,2)=31,            w = 0 at ζ = 0, 1.
 &end
 &num
  n1=81, n2=31, n3=61,                   Use an 81 × 31 × 61 mesh.
 &end
 &time
  idtmod=1,                              Recompute Δτ every time step.
  idtau=5, cfl=1.,                       Spatially varying Δτ.
  ntime=20,                              20 time steps
  ictest=4, eps=4*1.0e-6,                Use R_avg = 10^-6 as convergence criteria.
 &end
 &turb
  iturb=20,                              Turbulent flow, Chien model.
 &end
 &ic
 &end
EOD

ja -cslt
```

The JCL and input for the second run were similar to that for the first run. The only differences were:

1. The *touch* and *ln* commands for the restart files were changed to:

   ```
   touch rq5
   touch rx5
   ln rq4 fort.11
   ln rx4 fort.13
   ln rq5 fort.12
   ln rx5 fort.14
   ```

2. In namelist RSTRT, IREST was set equal to 2, since the previous run used the two-equation turbulence model.
3. In namelist BC, the value of JBCT1(1,1) was changed from 2 to 0, and the value of JBCT1(2,1) was changed from 12 to 10. This changed the boundary conditions on k and ε at the upstream boundary from $\partial k/\partial\xi = \partial\varepsilon/\partial\xi = 0$ to $k = k_{init}$ and $\varepsilon = \varepsilon_{init}$, where k_{init} and ε_{init} are the values of k and ε at the end of the previous run.
4. In namelist TIME, the parameter NTSEQ was added, and the parameters NTIME and CFL were changed, as follows:

   ```
   ntseq=3,                        Use 3 time step sequences.
   idtau=5, cfl=1.,5.,10.,         Spatially varying Δτ, CFL = 1, then 5, then 10.
   ntime=100,500,400,              100 time steps in first sequence, 500 in second,
                                       400 in third.
   ```

5. In the QSUB statements, the CPU time limit was raised to 14400.

The JCL and input for the third run were similar to that for the second r--. The only differences were:

1. The *touch* and *ln* commands for the restart files were changed to:

   ```
   touch rq6
   touch rx6
   ln rq5 fort.11
   ln rx5 fort.13
   ln rq6 fort.12
   ln rx6 fort.14
   ```

2. In namelist TIME, the parameter NTSEQ was omitted, and the parameters NTIME and CFL were changed, as follows:

   ```
   idtau=5, cfl=10.,               Spatially varying Δτ, CFL = 10.
   ntime=1500,                     1500 time steps.
   ```

Computed Results

For these calculations, the convergence criterion was that the average residual for each equation be less than 10^{-6}. However, both calculations were stopped before reaching this level of convergence when examination of several flow-related parameters indicated that the solution was no longer changing appreciably with time. The average residual at the end of the Baldwin-Lomax calculation ranged from 10^{-3} for the x-momentum equation to 3×10^{-5} for the continuity equation. For the Chien calculation the values were 3×10^{-4} for the x-momentum equation and 5×10^{-6} for the continuity equation. For both cases the residuals were continuing to drop when the calculations were stopped. A total of 4,000 iterations was used for the Baldwin-Lomax calculations, requiring 31,803 seconds of CPU time. An additional 2,520 iterations were used for the Chein calculations, requiring 22,048 seconds of CPU time.

In Figure 9.6, the computed flow field from the Chien calculation is shown in the form of total pressure contours at five stations through the duct. (The upstream and downstream straight sections are not shown.) As the flow enters the first bend, the boundary layer at the bottom of the duct initially thickens due to the locally adverse pressure gradient in that region. In an S-duct, the high pressure at the outside (bottom) of the first bend drives the low energy boundary layer toward the inside (top) of the bend, while the core flow responds to centrifugal effects and moves toward the outside (bottom) of the bend. The result is a pair of counter-rotating secondary flow vortices in the upper half of the cross-section. These secondary flows cause a significant amount of flow distortion, as shown by the total pressure contours.

In the second bend, the direction of the cross-flow pressure gradients reverses, making the pressure higher in the upper half of the cross-section. However, the flow enters the second bend with a vortex pattern already established. The net effect is to tighten and concentrate the existing vortices near the top of the duct, in agreement with classical secondary flow theory. The resulting horseshoe-shaped distortion pattern at the exit of the second bend is typical of S-duct flows.

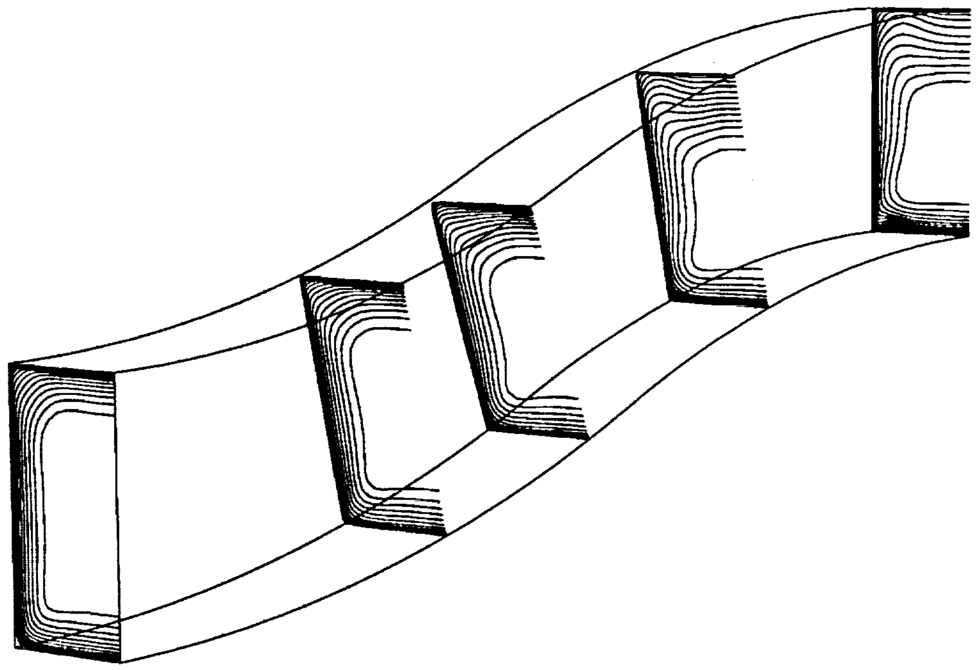

Figure 9.6. Computed total pressure contours for turbulent S-duct flow.

In Figure 9.7, the calculated wall pressure distribution is compared with the experimental data of Taylor, Whitelaw, and Yianneskis (1982). The agreement is very good. Both turbulence models correctly predicted the pressure trend and the pressure loss along the duct. The r and z coordinates noted in the legend are the same as those defined by Taylor, Whitelaw, and Yianneskis.

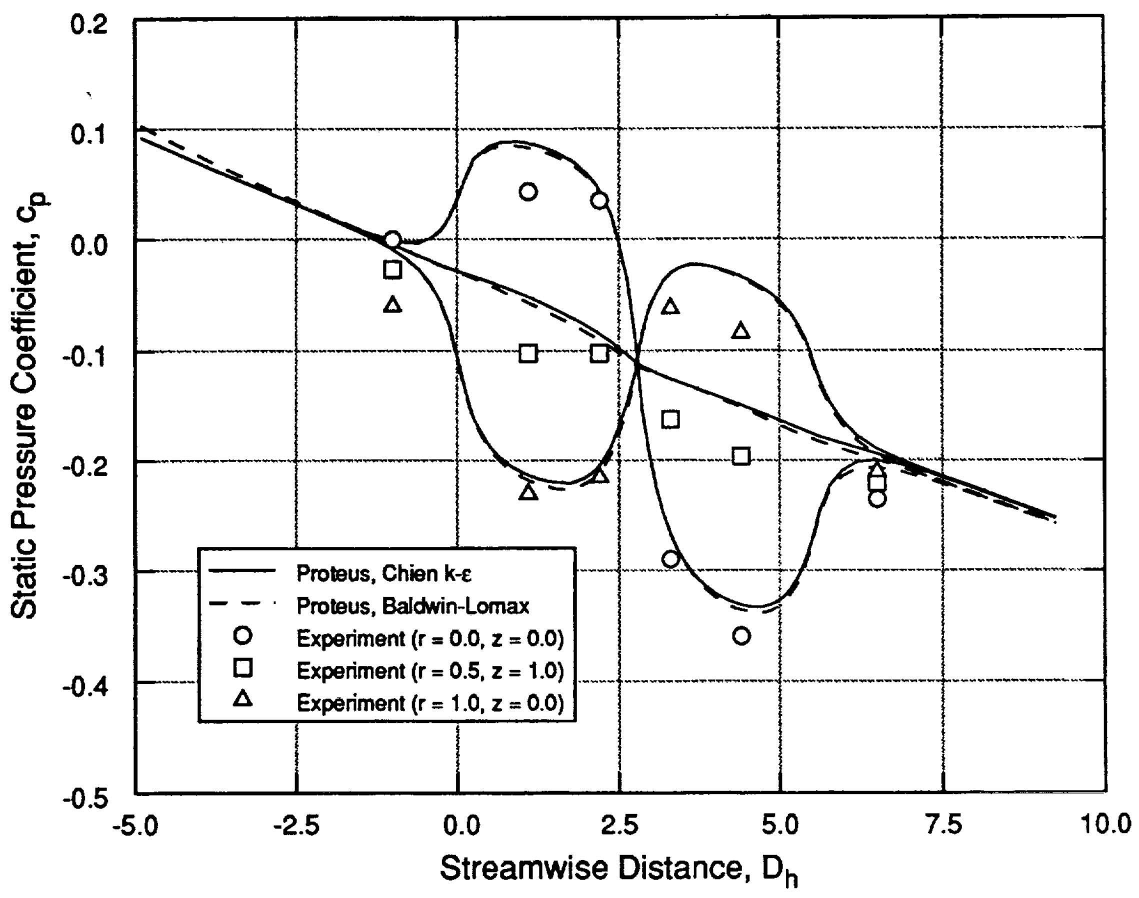

Figure 9.7. Computed surface static pressure distribution for turbulent S-duct flow.

In Figure 9.8, the experimental and calculated velocity profiles in the symmetry plane are shown for the five streamwise stations that were surveyed in the experiment. These surve·· stations are at the same locations as the total pressure contours shown in Figure 9.6. The agreement between computation and experiment is excellent for both turbulence models. The asymmetry in the velocity profiles due to the pressure induced secondary motion is correctly predicted.

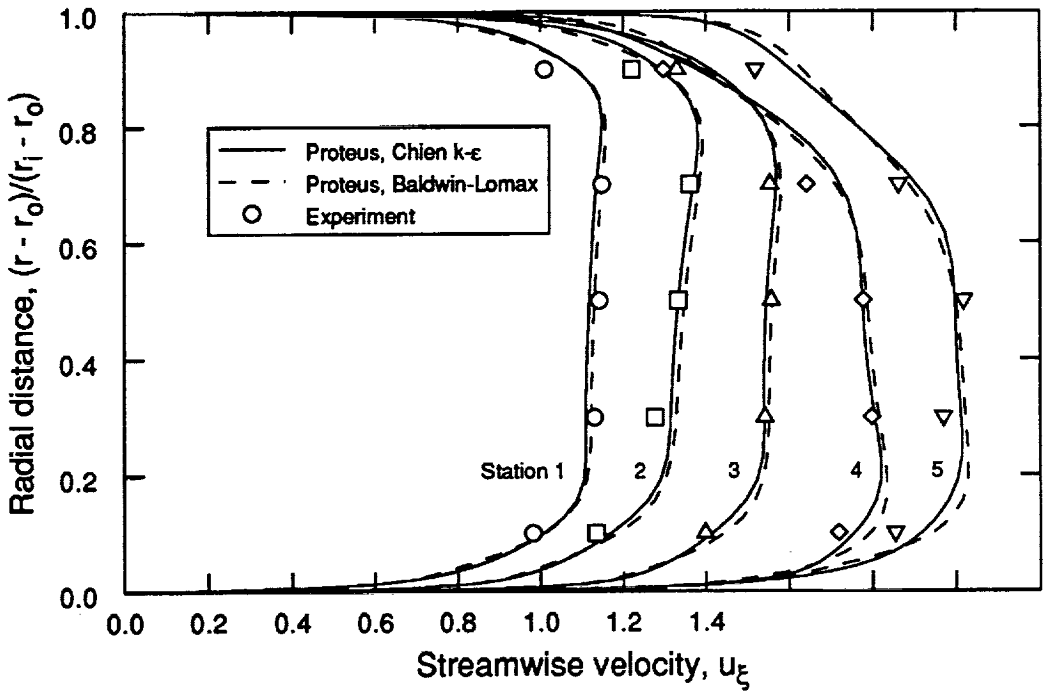

Figure 9.8. Computed streamwise velocity profiles for turbulent S-duct flow.

REFERENCES

Baldwin, B. S., and Lomax, H. (1978) "Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows," AIAA Paper 78-257.

Beam, R. M., and Warming, R. F. (1978) "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations," AIAA Journal, Vol. 16, No. 4, pp. 393-402.

Briley, W. R., and McDonald, H. (1977) "Solution of the Multidimensional Compressible Navier-Stokes Equations by a Generalized Implicit Method," Journal of Computational Physics, Vol. 24, pp. 373-397.

Cebeci, T., and Bradshaw, P. (1984) *Physical and Computational Aspects of Convective Heat Transfer,* Springer-Verlag, New York.

Chen, S. C., and Schwab, J. R. (1988) "Three-Dimensional Elliptic Grid Generation Technique with Application to Turbomachinery Cascades," NASA TM 101330.

Chien, K. Y. (1982) "Prediction of Channel and Boundary-Layer Flows with a Low-Reynolds-Number Turbulence Model," AIAA Journal, Vol. 20, No. 1, pp. 33-38.

Cooper, G. K. (1987) "The PARC Code: Theory and Usage," AEDC-TR-87-24.

Cray Research, Inc. (1988) *UPDATE Reference Manual,* Publication Number SR-0013.

Cray Research, Inc. (1989a) *Volume 1: UNICOS Fortran Library Reference Manual,* Publication Number SR-2079.

Cray Research, Inc. (1989b) *Volume 3: UNICOS Math and Scientific Library Reference Manual,* Publication Number SR-2081.

Cray Research, Inc. (1990) *CF77 Compiling System, Volume 1: Fortran Reference Manual,* Publication Number SR-3071.

Cray Research, Inc. (1991) *UNICOS User Commands Reference Manual,* Publication Number SR-2011.

Dongarra, J. J., Moler, C. B., Bunch, J. R., and Stewart, G. W. (1979) *LINPACK User's Guide* SIAM, Philadelphia.

Hughes, W. F., and Gaylord, E. W. (1964) *Basic Equations of Engineering Science,* Schaum's Outline Series, McGraw-Hill Book Company, New York.

Jameson, A., Schmidt, W., and Turkel, E. (1981) "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes," AIAA Paper 81-1259.

Kernighan, B. W., and Plauger, P. J. (1978) *The Elements of Programming Style,* McGraw-Hill Book Company, New York.

Kleinstein, G. (1967) "Generalized Law of the Wall and Eddy-Viscosity Model for Wall Boundary Layers," AIAA Journal, Vol. 5, No. 8, pp. 1402-1407.

Knight, C. J., and Choi, D. (1989) "Development of a Viscous Cascade Code Based on Scalar Implicit Factorization," AIAA Journal, Vol. 27, No. 5, pp. 581-594.

Launder, B. E., and Priddin, C. H. (1973) "A Comparison of Some Proposals for the Mixing Length Near a Wall," International Journal of Heat and Mass Transfer, Vol. 16, pp. 700-702.

Nichols, R. H. (1990) "Two-Equation Model for Compressible Flows," AIAA Paper 90-0494.

Roberts, G. O. (1971) "Computational Meshes for Boundary Layer Problems," Proceedings of the Second International Conference on Numerical Methods in Fluid Dynamics, *Lecture Notes in Physics,* Vol. 8, Springer-Verlag, New York, pp. 171-177.

Spalding, D. B. (1961) "A Single Formula for the Law of the Wall," Journal of Applied Mechanics, Vol. 28, pp. 455-457.

Sparrow, E. M., Hixon, C. W., and Shavit, G. (1967) "Experiments on Laminar Flow Development in Rectangular Ducts," Transactions of the ASME, Journal of Basic Engineering, Vol. 89, pp. 116-124.

Steger, J. L. (1978) "Implicit Finite-Difference Simulation of Flow about Arbitrary Two-Dimensional Geometries," AIAA Journal, Vol. 16, No. 7, pp. 679-686.

Steinbrenner, J. P., Chawner, J. P., and Fouts, C. L. (1991) "The GRIDGEN 3D Multiple Block Grid Generation System, Volume II: User's Manual," WRDC-TR-90-3022, Vol. II.

Taylor, A. M. K. P., Whitelaw, J. H., and Yianneskis, M. (1982) "Developing Flow in S-Shaped Ducts," NASA CR 3550.

Towne, C. E., Schwab, J. R., Benson, T. J., and Suresh, A. (1990) "PROTEUS Two-Dimensional Navier-Stokes Computer Code - Version 1.0, Volumes 1-3," NASA TM's 102551-3.

Walatka, P. P., Buning, P. G., Pierce, L., and Elson, P. A. (1990) "PLOT3D User's Manual," NASA TM 101067.

Wassel, A. T., and Catton, I. (1973) "Calculation of Turbulent Boundary Layers Over Flat Plates With Different Phenomenological Theories of Turbulence and Variable Turbulent Prandtl Number," International Journal of Heat and Mass Transfer, Vol. 16, pp. 1547-1563.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE October 1993	3. REPORT TYPE AND DATES COVERED Technical Memorandum

4. TITLE AND SUBTITLE Proteus Three-Dimensional Navier-Stokes Computer Code–Version 1.0 Volume 2–User's Guide	5. FUNDING NUMBERS WU–505–62–52
6. AUTHOR(S) Charles E. Towne, John R. Schwab, and Trong T. Bui	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135–3191	8. PERFORMING ORGANIZATION REPORT NUMBER E–8109
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, D.C. 20546–0001	10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA TM–106340

11. SUPPLEMENTARY NOTES

Responsible person, Charles E. Towne, (216) 433–5851.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified - Unlimited Subject Category 34	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A computer code called *Proteus 3D* has been developed to solve the three-dimensional, Reynolds-averaged, unsteady compressible Navier-Stokes equations in strong conservation law form. The objective in this effort has been to develop a code for aerospace propulsion applications that is easy to use and easy to modify. Code readability, modularity, and documentation have been emphasized. The governing equations are solved in generalized nonorthogonal body-fitted coordinates, by marching in time using a fully-coupled ADI solution procedure. The boundary conditions are treated implicitly. All terms, including the diffusion terms, are linearized using second-order Taylor series expansions. Turbulence is modeled using either an algebraic or two-equation eddy viscosity model. The thin-layer or Euler equations may also be solved. The energy equation may be eliminated by the assumption of constant total enthalpy. Explicit and implicit artificial viscosity may be used. Several time step options are available for convergence acceleration. The documentation is divided into three volumes. This is the User's Guide, and describes the program's features, the input and output, the procedure for setting up initial conditions, the computer resource requirements, the diagnostic messages that may be generated, the job control language used to run the program, and several test cases.

14. SUBJECT TERMS Navier-Stokes; Computational fluid dynamics; Viscous flow; Compressible flow	15. NUMBER OF PAGES 104
	16. PRICE CODE A06

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT